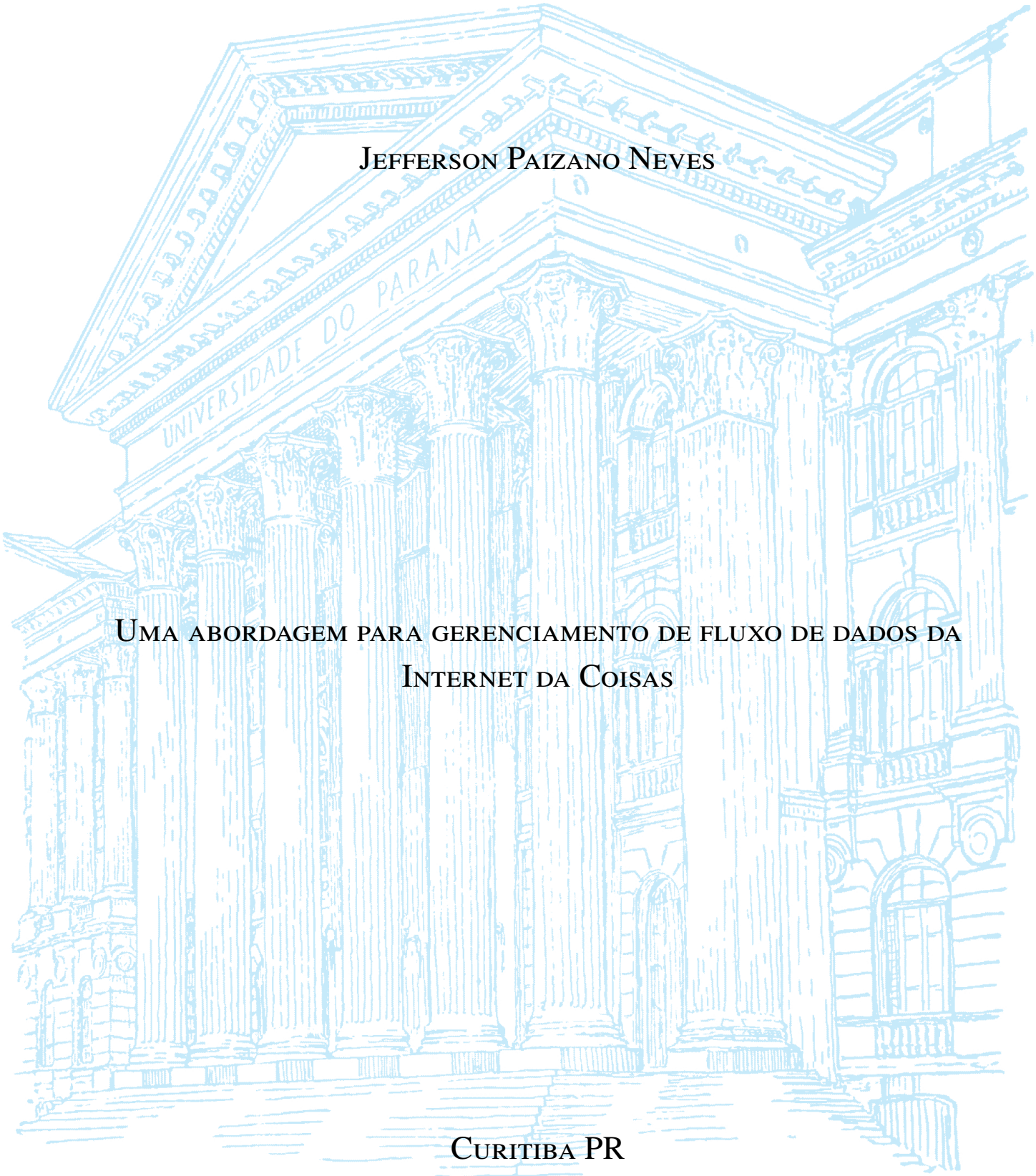
Universidade Federal do Paraná

Jefferson Paizano Neves



Uma abordagem para gerenciamento de fluxo de dados da Internet da Coisas

Curitiba PR
2017

Jefferson Paizano Neves

Uma abordagem para gerenciamento de fluxo de dados da Internet da Coisas

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Luis Carlos Erpen de Bona.

Curitiba PR
2017

N513a Neves, Jefferson Paizano
Uma abordagem para gerenciamento de fluxo de dados da internet das coisas / Jefferson Paizano Neves. – Curitiba, 2017.
80 p. : il. color. ; 30 cm.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2017.

Orientador: Luis Carlos Erpen de Bona .
Bibliografia: p. 73-80.

1. Fluxo de dados (Computadores). 2. Processamento eletrônico de dados. 3. Internet. I. Universidade Federal do Paraná. II.Bona, Luis Carlos Erpen de. III. Título.

CDD: 004.68

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
Setor CIÊNCIAS EXATAS
Programa de Pós-Graduação INFORMÁTICA

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **JEFFERSON PAIZANO NEVES** intitulada: **Uma abordagem para gerenciamento de fluxo de dados da Internet das Coisas**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 14 de Setembro de 2017.

LUIS CARLOS ERPEN DE BONA
Presidente da Banca Examinadora (UFPR)

ANDRÉ RICARDO ABED GRÉGIO
Avaliador Interno (UFPR)

GUILHERME GALANTE
Avaliador Externo (UNIOESTE)

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

Dedico esta dissertação à minha filha querida, Ana Cecília.

Agradecimentos

Em primeiro lugar, a Deus.

Aos meus pais, Rubens J. Neves e Suzely P. Neves, por todo incentivo e apoio nesta jornada. À minha esposa, Flávia L. Ojeda, pela compreensão, desde o momento do ingresso no mestrado. À minha irmã, Jeisy C. A. Paizano Neves, pela disponibilidade em cuidar de minha filha. À todos os meus familiares que apoiaram nesta jornada. À minha sogra e a bisa, Dulsangelica Leal e Anastácia Leal, por todo zelo e carinho ao cuidar da minha filha durante esse período que estive ausente.

Ao meu orientador, Prof. Luis Bona, pela orientação, paciência e, principalmente, por aceitar um orientando que já estava com o mestrado em andamento.

Aos colegas e amigos de laboratório Benevid Felix, Danilo Faria, Alison Puska, Adi Marcondes e, em especial Otto Pinno e Daniel Rezende por terem ajudado na revisão e correção da dissertação de mestrado. Aos colegas e amigos de Dinf Alexandre Alencar, Paulo Lacerda, Ivan Pires, Valber Zacarkim, Santiago Viertel e Cides Bezerra.

Resumo

O avanço das tecnologias de comunicação e a redução dos dispositivos computacionais estão propiciando o desenvolvimento da Internet das Coisas (IoT). Esta rede integra desde geladeiras, relógios, bicicletas até dispositivos computacionais, promovendo a interação entre os objetos e os seres humanos em ambientes como domiciliares e industriais. Com a integração da comunicação entre os objetos, dispositivos computacionais e seres humanos, vários serviços personalizados poderão ser prestados em tempo real, tais como o monitoramento das funções vitais, operações de gerenciamento de emergências e mensuração da temperatura. Contudo, a grande quantidade de objetos inseridos nos ambientes produzem em velocidades elevadas um enorme volume de dados. Esta gama de informações geradas coloca a IoT como um ponto chave no desenvolvimento de aplicações de análise e processamento de dados. No entanto, aspectos relacionados a processamento, encaminhamento e armazenamento dos dados precisão ser revisados e aperfeiçoados para as aplicações da IoT, pois, o tráfego dos dados para os provedores de serviços em nuvem pode degradar os canais de comunicação e aumentar a latência nos serviços. Por este motivo, o conceito emergente da computação em Névoa (Fog) surgiu no intuito de utilizar os recursos disponíveis próximo do usuário final para executar o processamento, encaminhamento e armazenamento dos dados gerados na IoT. Assim, a *Fog* permite manter o tráfego dos dados e o processamento na borda da rede. Este trabalho propõe uma abordagem para gerenciamento de fluxo de dados da Internet das Coisas na *Fog*. Essa abordagem utiliza a *Fog* em conjunto com uma plataforma para IoT, de modo a reduzir o tráfego de dados brutos e a latência na rede. Desta forma, os dados brutos são processados na borda da rede e encaminhados direto ao nó consumidor e/ou para a infraestrutura da Nuvem. A abordagem foi avaliado através de simulações que mostraram sua eficácia no gerenciamento dos dados na borda da rede, reduzindo a latência dos serviços.

Palavras-chave: IoT, Fog, Gerenciamento de fluxo de dados, Plataforma para IoT.

Abstract

The advancement of communication technologies and the reduction of computing devices are enabling the development of the Internet of Things (IoT). This network integrates from refrigerator, clock, bikes to computing devices, promoting the interaction between objects and humans in environments such as home and industrial. With the integration of communication between objects, computing devices, and many custom services humans can provided in real-time as the monitoring of vital functions, emergency management operations, measurement of temperature. However, the large amount of inserted objects in the environments at high speeds produce a huge volume of data. This range of information generated puts the IoT as a key point in the development of analysis and data processing applications. However, aspects related to data processing, routing, and storage need to be reviewed and improved for IoT applications because data traffic to cloud service providers can degrade communication channels and increase service latency. For this reason, the emergent concept of Fog Computing has appeared to use the available resources near the end-user to do the processing, routing and storage of the data generated in IoT. Fog allows you to keep data traffic and processing at the edge of the network. This paper proposes an approach to data flow management the Internet of Things in Fog. This approach uses Fog with an IoT platform to cut raw data traffic and latency on the network. In this way, the raw data is process at the edge of the network and send directly to the consumer node and/or to the cloud infrastructure. The approach was test through simulations that showed its effectiveness in data management at the edge of the network, reducing service latency.

Keywords: IoT, Fog, Data flow management, Platform for IoT.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

AF	Agente *Fog*
CC	Computação em Nuvem
CoAP	*Constrained Application Protocol*
DaaS	*Data-as-a-Service*
Fog	Computação em Névoa
GE	*Generic Enablers*
IaaS	*Infrastructure-as-a-Service*
IETF	*Internet Engineering Task Force*
IoT	*Internet of Things*
MAC	*Media Access Control*
MQTT	*MQ Telemetry Transport*
M2M	*Machine-to-Machine*
NFC	*Near Field Communication*
NGSI	*Next Generation Services Interface*
PaaS	*Platform-as-a-Service*
RDF	*Resource Description Format*
REST	*Representational State Transfer*
RFID	*Radio Frequency Identification*
RPL	*Routing Protocol for Low-Power and Lossy*
RSSF	Redes de Sensores Sem Fio
SaaS	*Sensing-as-a-Service*
SDN	Redes Definidas por *Software*
SenaaS	*Sensor-as-a-Service*
SOA	*Service-Oriented Architecture*
XMPP	*Extensible Messaging and Presence Protocol*
6LowPAN	*IPv6 over Low Power Wireless Personal Area Network*

Capítulo 1

Introdução

A Internet atual está passando por sua próxima evolução. Essa evolução permitirá que as coisas do nosso cotiano como geladeiras, relógios e roupas sejam incorporadas à Internet, surgindo a Internet das Coisas (IoT do acrônimo em inglês *Internet of Things*) [Perera et al., 2014]. A IoT enfatiza a interconexão de coisas por meio redes de curto alcance. Com essa interconexão, as coisas poderão interagir entre si e/ou com dispositivos computacionais, proporcionando serviços personalizados e eficientes às pessoas. Dessa maneira, o desenvolvimento da IoT é considerado a próxima onda na indústria global da Internet [Chunli, 2012].

O paradigma da IoT resume-se em uma rede heterogênea que integra coisas à Internet [Gubbi et al., 2013]. Esta integração proporciona uma mudança tanto na comunicação quanto nos serviços oferecidos via Internet. A troca de informação centra-se na comunicação de máquina-para-máquina (M2M do acrônimo em inglês *machine-to-machine*). A comunicação entre as coisas permitem a coleta e/ou compartilhamento de informações do mundo real [Vasseur e Dunkels, 2010]. Assim, a cooperação entre as coisas permitem o desenvolvimento de serviços personalizados como, mensuração de temperatura, monitoramento de consumo elétrico e localização geográfica [Miorandi et al., 2012]. Além disso, através da IoT, serviços mais complexos podem ser implementados como, operações de gerenciamento de emergências [Yang et al., 2013].

A IoT pode ser inserida em vários ambientes do domínio urbano como doméstico, saúde, transporte e industrial. Com a IoT, esses ambientes tornam-se inteligentes, pois, os dados gerados pelas coisas podem agregar conhecimento útil na tomada de decisões [Atzori et al., 2010]. Por exemplo, as coisas contidas no ambiente da saúde podem coletar dados vitais de pacientes em tempo real, em ambientes industriais auxiliam em processos de produção, no ambiente de transporte podem melhorar a logística na entrega de mercadorias e no ambiente doméstico contribuem para o conformo e segurança das pessoas. Logo, a adoção da IoT contribui para o desenvolvimento de cidades inteligentes.

O avanço das tecnologias de redes sem fio proporcionou o desenvolvimento da IoT. A grande maioria das coisas possuem limitações de energia, processamento, armazenamento e enlaces sem fio com perdas em comunicações multi-salto. Devido as estas restrições, os padrões utilizados em outras redes não se adéquam às necessidades de comunicação da IoT. Desta maneira, a IoT utiliza o padrão IEEE 802.15.4 [Khanafer et al., 2014] e 6LoWPAN [Sheng et al., 2013] na comunicação entre as coisas. O padrão IEEE 802.15.4 especifica a camada física e o controle de acesso para redes de área pessoal com baixas taxas de transmissão. A tecnologia 6LoWPAN permite, com base no protocolo IPv6, a integração do grande número de coisas existente na IoT. Portanto, estas tecnologias possibilitam a identificação e a troca de informações na IoT.

A integração das de coisas da IoT causa um aumento no volume de dados gerados. Na literatura estima-se que em 2020 a IoT terá 20 bilhões de coisas conectadas à rede [Gubbi et al., 2013]. Essa ampla quantidade de coisas irá gerar, em velocidades elevadas, um grande volume de dados. Esta gama de informação gerada coloca a IoT como um ponto chave no desenvolvimento de aplicações de análise e processamento de dados. No entanto, a transmissão desses dados podem consumir muitos recursos da rede. Assim, aspectos relacionados a armazenamento, análise de fluxo de dados, e processamento distribuídos precisem ser revisados e aperfeiçoados em aplicações da IoT.

Um dos pontos chave da IoT é a análise de dados gerados por coisas contidas nos ambientes inteligentes. No entanto, gerenciar esses dados torna-se cada vez mais desafiador devido ao grande volume de dados gerados por todas as coisas do mundo real. O aumento no volume e na velocidade de dados gerados pode consumir os recursos de comunicação da rede e piorar requisitos como a latência. Além disso, os tipos de dados gerados pela IoT, por exemplo, fluxos de vídeo e medições em tempo real, aumentam o estresse sobre a infraestrutura de rede. Desta forma, técnicas para filtrar, encaminhar e processar os dados na borda da rede são cruciais para os ambientes da internet das coisas, pois, efetuam a redução do tráfego injetado na infraestrutura de comunicações.

As abordagens que atualmente contribuem com o gerenciamento de dados da IoT são geralmente baseadas nos conceitos de CC (Computação em Nuvem), Fog (Computação em Névoa) ou SDN (Rede Definida por Software). A CC permite acesso ubíquo e sob demanda a um conjunto de recursos computacionais elásticos e configuráveis como, rede, servidores, armazenamento, aplicações e serviços [Mell e Grance, 2011]. A *Fog* fornece parte dos recursos existentes no núcleo para a borda rede, ou seja, fornece processamento, armazenamento e serviços de rede entre o dispositivo do usuário final e os servidores tradicionais da computação em nuvem [Bonomi et al., 2012]. A SDN centraliza o plano de controle em controladores baseados em *software* e transforma o dispositivo de rede em um encaminhador de pacotes [Farhady et al., 2015].

Na literatura existem diferentes soluções que buscam gerenciar os dados da IoT. Um grande número de projetos buscam desenvolver plataformas baseadas em CC que gerencie, processe, armazene e ofereça como serviço os dados gerados pela IoT [Mineraud et al., 2016]. Os projetos Fiware [Fiware, 2017b] e OpenIoT [OpenIoT, 2017] são exemplos de plataformas que integram as coisas da IoT com a CC. Os trabalhos [Gazis et al., 2015, Aazam e Huh, 2014] utilizam infraestrutura da *Fog* para executar o pré-processamento dos dados da IoT antes de os encaminhar para a nuvem. Em [Wu et al., 2015, Qin et al., 2014] são utilizados os conceitos de SDN para minimizar os esforços de gerenciamento tanto da rede quanto do fluxo de dados.

Este trabalho apresenta uma abordagem para o gerenciamento de fluxo de dados da IoT na *Fog* que busca reduzir a latência e o consumo excessivo dos recursos de rede. Ela provê um filtro de dados definido por *software* na *Fog*. A abordagem utiliza a infraestrutura da plataforma baseada em Nuvem para a IoT do projeto Fiware [Fiware, 2017b] em conjunto com os conceitos de *Fog* para filtrar e encaminhar os dados gerados em ambientes da internet das coisas. Desta forma, os dados brutos são processados na borda da rede e encaminhados direto ao nó consumidor e/ou para a infraestrutura da nuvem. A abordagem foi avaliado através de simulações que mostraram sua eficácia no gerenciamento dos dados na borda da rede, reduzindo a carga e a latência na rede.

Esta proposta está organizada em quatro capítulos. O Capítulo 2 apresenta os fundamentos relacionados à IoT, suas características, seu funcionamento e os problemas. O Capítulo 3 apresenta algumas abordagens que buscam solucionar gerenciar os dados gerados na IoT. No Capítulo 4 apresenta-se a abordagem para o gerenciamento de fluxo de dados da IoT na Fog, a

sua estrutura e o seu funcionamento. O Capítulo 5 apresenta o ambiente utilizado para efetuar a simulação e os resultados obtidos durante os experimentos. Por fim, no Capítulo 6 apresentam-se as conclusões e discute possíveis trabalhos futuros.

Capítulo 2

Internet das Coisas

O conceito de Internet das Coisas (IoT, do inglês *Internet of Things*) descreve uma nova evolução na comunicação e computação na internet atual. A Figura 2.1 ilustra de maneira simplificada esta evolução, que no início buscava conectar somente computadores. Em seguida, a internet móvel surge da incorporação de dispositivos móveis a internet. E agora, a integração de coisas com a Internet a coloca na direção da sua próxima evolução, denominada Internet das Coisas. Esse conceito foi apresentado inicialmente por Mark Weiser no início de 1990 [Weiser, 1991]. No entanto, o termo Internet das Coisas se tornou popular após a publicação do trabalho *"Internet of Things"* desenvolvido pelo centro de pesquisa Auto-ID localizado no Instituto Tecnológico de Massachusetts (MIT, do Inglês *Massachusetts Institute of Technology*) em 1999 [Sarma et al., 2000]. Com sua popularização, a IoT foi considerada a terceira onda na indústria global da internet [Chunli, 2012].

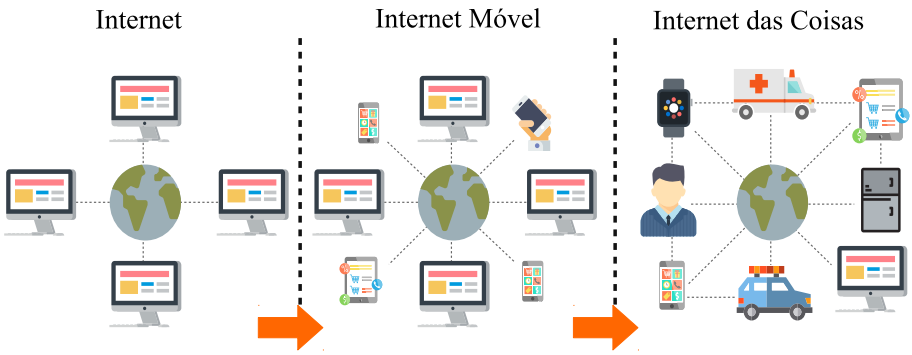


Figura 2.1: Evolução da Internet, inspirado em [Perera et al., 2014].

O paradigma da IoT proporciona a integração de coisas em redes híbridas, abertas e heterogêneas conectadas à Internet [Gubbi et al., 2013]. Esta integração proporciona uma mudança tanto na comunicação quanto nos serviços oferecidos via internet. As coisas contidas neste conceito resumem-se em qualquer objeto físico como, por exemplo, lâmpadas, geladeiras e até roupas, que seja capaz de coletar e/ou transmitir informações [Vasseur e Dunkels, 2010]. Essas propriedades permitem aos objetos coletar informações de um ambiente e as compartilharem entre si e/ou com dispositivos computacionais. Assim, esta cooperação proporciona o desenvolvimento de serviços personalizados.

A interação entre os objetos na IoT possibilita o desenvolvimento de um grande número de serviços personalizados. A Tabela 2.1 apresenta alguns domínios como, transporte e logística, saúde, ambiente residencial, pessoal e social, e possíveis aplicações que poderiam ser desenvolvidas nesses domínios. O transporte e logística englobam o controle de tráfego através

do sensoriamento de estradas, de veículos tanto de passeio quanto de transporte de mercadorias [He et al., 2014]. No domínio médico podemos ressaltar o monitoramento de pacientes, obtenção automática de informações e identificação dos pacientes [Fernandez e Pallis, 2014]. Em ambientes residenciais as aplicações poderiam controlar temperatura, a iluminação ou sistemas de alerta para evitar incidentes [Gubbi et al., 2013]. As aplicações no domínio pessoal e social possibilitam a interação pessoal e meios para a construção de novos relacionamentos sociais. As aplicações de domínio futurístico poderiam oferecer serviços como de táxi robô, onde criaria um serviço de táxi que se adéqua em tempo real aos movimentos de tráfego. Desta forma, o paradigma da IoT permite agregar inteligência aos ambientes de cada domínio apresentado.

Domínio	**Aplicação**
Transporte e Logística	Sensoriamento Ambiental
	Logística
	Controle de Tráfego
Saúde	Monitoramento
	Coleta de Dados
	Identificação e Autenticação
Ambiente Residencial	Conforto Domiciliar
	Monitoramento de Incidentes
Pessoal e Social	Interação Pessoal
	Relacionamento Social
Futurístico	Táxi Robô

Tabela 2.1: Domínios de aplicação da IoT, adaptado de [Atzori et al., 2010].

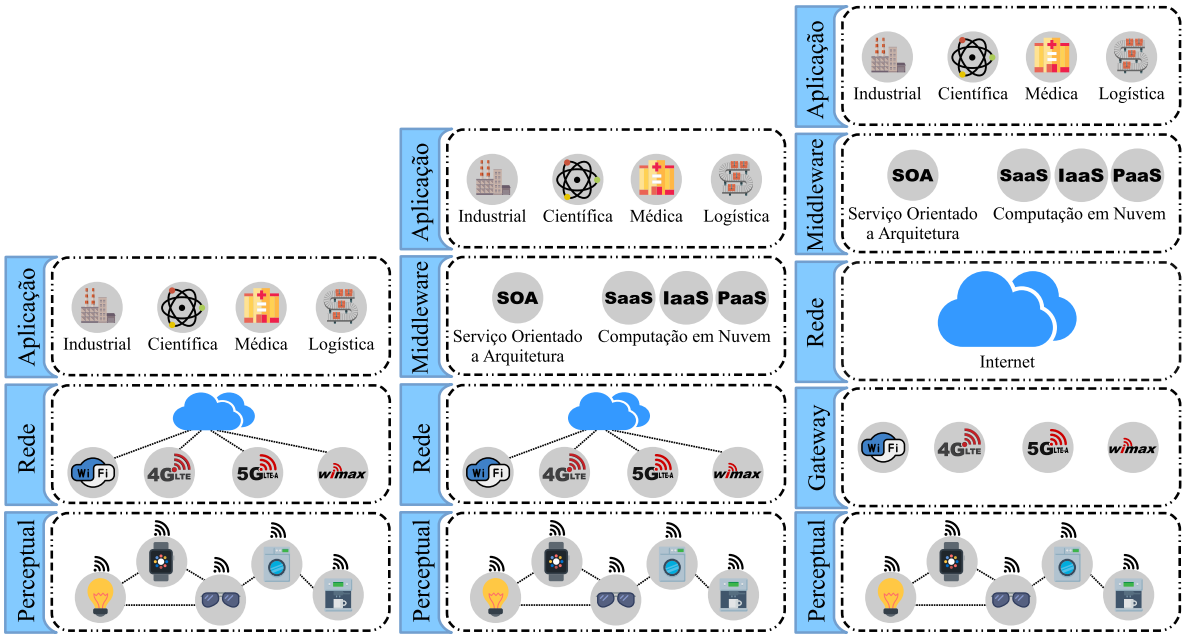
Um ambiente torna-se inteligente quando obtêm conhecimento útil a partir de dados brutos coletados por objetos [Atzori et al., 2010]. Seu funcionamento se divide em ambiente físico e computação virtual [Whitmore et al., 2015]. O ambiente físico resume-se em uma área do mundo real que possui em sua extensão, conjuntos de sensores e dispositivos computacionais para a coleta de informações como, temperatura, úmida, etc. A computação virtual proporciona o processamento, encaminhamento e análise dos dados coletados por sensores localizados no ambiente físico. Assim, devido ao seu conceito de interconexão de coisa do mudo real, os ambientes inteligentes da IoT podem gerar um grande volume de dados.

Na literatura estima-se que até 2020 a IoT terá em torno de 20 bilhões de coisas conectadas à rede [Gubbi et al., 2013]. Essa ampla quantidade de coisas produz, em velocidades elevadas, um grande volume de dados. Além disso, as coisas possuem capacidade de produzir sequências potencialmente ilimitadas de dados, denominadas de fluxo de dados [Silva et al., 2013]. Esta gama de informação gerada coloca a IoT como um ponto chave no desenvolvimento de aplicações de análise e processamento de dados. No entanto, características como, variação no tempo de geração, taxas de amostragem distintas e formatos não uniformes de dados, fazem com que aspectos relacionados a armazenamento, análise de fluxos de dados, e processamento distribuído precisem ser revisados e aperfeiçoados em aplicações da IoT.

O restante do capítulo apresenta os fundamentos necessários à compreensão do contexto da pesquisa, do problema e da proposta. A Seção 2.1 apresenta sugestões de arquiteturas para a IoT, bem como as tecnologias contidas na rede e as técnicas de comunicação existentes nesta evolução da internet. A Seção 2.2 apresenta uma visão centrada nos dados da IoT, além de, descrever as principais caraterísticas dos dados gerados pela internet das coisas. A Seção 2.3 mostra um resumo do conteúdo abordado no capítulo.

2.1 Arquitetura

As organizações responsáveis pelo processo de padronização da internet das coisas ainda não estabeleceram um padrão para sua arquitetura [Minerva et al., 2015]. Esta ausência de padronização contribuiu para o surgimento de diferentes modelos de arquitetura na literatura, por exemplo, os modelos com três [Amaral et al., 2016], quatro [Xu et al., 2014] e cinco [Aazam et al., 2014, Khan et al., 2012] camadas. Esses modelos provem meios para a realização da comunicação entre os objetos e as aplicações [Li et al., 2015]. A comunicação deve satisfazer requisitos como interoperabilidade e escalabilidade. A interoperabilidade garante a comunicação transparente de sistemas e dispositivos que utilizam diferentes tipos de tecnologias. A escalabilidade possibilita o gerenciamento do crescente número de dispositivos na IoT. Assim, devido à existência de grande heterogeneidade na rede, esses requisitos possuem papel importante no desenvolvimento de aplicações e serviços na IoT. A Figura 2.2 apresenta as arquiteturas em três, quatro e cinco camadas.



(a) Três Camadas (b) Quatro Camadas (c) Cinco Camadas

Figura 2.2: Arquiteturas da IoT

Os três modelos de arquitetura possuem semelhanças em sua estrutura. Essa semelhança ocorre na existência das camadas perceptual ou sensitiva, de rede ou transporte e aplicação em todos os modelos. A arquitetura em três camadas, mostrada na Figura 2.2(a), se divide em camada perceptual ou sensitiva, de rede ou transporte e aplicação [Amaral et al., 2016]:

Camada Perceptual ou Sensitiva: nessa camada estão inseridos os dispositivos de sensoriamento do ambiente físico. Estes dispositivos utilizam tecnologias como RFID [Jia et al., 2012], RSSF [Rawat et al., 2014], *Zigbee* [Gungor e Hancke, 2009], e *Bluetooth* [Chang, 2014]. O RFID fornece uma identificação única e em tempo real para os objetos. Os sensores efetuam a coleta de dados como temperatura, movimento, umidade e alterações químicas do ambiente físico. As tecnologias *ZigBee* e *Bluetooth* definem um padrão de comunicação de curta distância com baixo consumo de energia para os dispositivos. Desta forma, por ser a origem dos dados gerados pelo ambiente físico, a camada perceptual é considera a base da IoT.

Camada de Rede ou Transporte: responsável pela transferência das informações coletadas para as aplicações. Ela recolhe os dados a partir da camada inferior e os envia via rede de acesso à Internet. O envio desses dados para Internet ocorre através de redes de longa distância como LTE, LTE-A [Cao et al., 2014] e Satélite ou por redes *Wi-fi* [Ferro e Potorti, 2005]. Além disso, esta camada deve gerenciar e controlar a grande quantidade de dados gerados pela IoT.

Camada Aplicação: disponibiliza as informações coletadas para o usuário final da IoT. Esta camada tem como objetivo integrar todas as funções das camadas inferiores e disponibilizar serviços para os seres humanos. A partir desses serviços o usuário final pode efetuar o processamento, a análise, e a filtragem das informações geradas pelos dispositivos da IoT. Desta forma, esta camada fornece funções essenciais para o desenvolvimento de aplicações para o bem da população, e, como tal, gerar melhoria na saúde, no transporte, na logística, na indústria, entre outras.

O modelo de quatro camadas dispõe de uma camada adicional que diverge do modelo anterior. Como pode ser visto na Figura 2.2(b), no modelo em quatro camadas possui a camada de *middleware* [Xu et al., 2014]:

Camada de Middleware: busca prover uma flexibilidade na associação entre interfaces de *hardware* e *software*. Os dispositivos da IoT utilizam diferentes tecnologias de comunicação e podem fornecer diversos serviços. Desta forma, esta grande heterogeneidade necessita de uma plataforma de *middleware* para garantir o gerenciamento dos serviços, o armazenamento dos dados, e controle dos dispositivos. Esta plataforma pode ser SOA (*Service-Oriented Architecture*) ou serviços de computação em nuvem como, por exemplo, SaaS (*Software-as-a-service*), IaaS (*Infrastructure-as-a-Service*), e PaaS (Platform-as-a-Service).

Na arquitetura em cinco camadas existe uma camada adicional que diverge do modelo em quatro camadas. Como pode ser visto na Figura 2.2(c), no modelo em cinco camadas possui a camada de *gateway* [Aazam et al., 2014]:

Camada de Gateway: responsável por ser o intermediário entre a camada perceptual e a de rede. A camada de *gateway* controla a comunicação dos objetos inteligentes com a internet. Este controle ocorre em redes de acesso à internet como, por exemplo, redes LTE [Cao et al., 2014], *Wi-Fi* [Ferro e Potorti, 2005] ou outros tipos de redes de acesso.

A camada perceptual ou sensitiva, existente em todos os modelos, é considera a base da IoT. Esta importância está diretamente ligada na identificação e integração das coisas existentes nas redes da IoT. Além disso, ele possibilita a coleta dos dados gerados por cada coisa. Assim, essa camada é a fonte de dados necessária para o desenvolvimento de aplicações e serviços baseados na IoT.

2.1.1 Camada Perceptual ou Sensitiva

As tecnologias contidas na IoT variam de acordo com a finalidade do dispositivo e o tipo de sensoriamento necessário em determinado ambiente inteligente. Geralmente, os dispositivos responsáveis pelo sensoriamento do mundo físico utilizam tecnologias como Identificação por Radiofrequência (RFID) [Jia et al., 2012], Comunicação de Campo Próximo (NFC) [Coskun et al., 2013], e Redes de Sensores Sem Fio (RSSF) [Rawat et al., 2014]. Desta

forma, a adoção dessas tecnologias possibilita a implementação e a evolução do paradigma da IoT.

A RFID (*Radio Frequency Identification*) é uma tecnologia de comunicação sem fio de identificação automática. As redes RFID possuem em sua composição, dispositivos leitores e as etiquetas. O dispositivo leitor RFID efetua a leitura das informações de identificação contidas nas etiquetas. No que lhe concerne, as etiquetas RFID armazenam e respondem as requisições feitas pelo dispositivo leitor. As etiquetas de redes RFID se dividem em duas categorias, sendo elas, passiva e ativa [Xu et al., 2014]. A primeira categoria resume-se em pequenos *microchips* de silício com antenas que utiliza radiofrequência do leitor para transmitir seus dados. Ao contrário da passiva, a etiqueta ativa não precisa utilizar a radiofrequência do dispositivo leitor para transmitir seus dados, pois, conta com uma bateria própria para transmitir o sinal. Assim, devido ao tamanho pequeno e o longo tempo de vida útil das etiquetas, as redes RFID podem contribuir para o rastreio e a identificação de objetos, produtos ou até mesmo indivíduos em diversos ambientes da IoT.

A NFC (*Near Field Communication*) permite uma comunicação de curto alcance e de forma segura entre dispositivos. Esta comunicação ocorre automaticamente entre os objetos quando se aproximam o suficiente um do outro. A troca de informação na NFC dá-se em faixa de frequência 13.56MHz com velocidades de transmissão de 106 Kbits/s, 212 Kbits/s e 424 Kbits/s. A NFC opera em modo ativo e passivo com alcance médio de 10 cm. O modo ativo, ombos os dispositivos geram sinal de radiofrequência para transmitir seus dados. Em modo passivo, apenas um dispositivo gera o sinal de conexão. Esta tecnologia pode ser aplicada em métodos de pagamento, emissão de bilhetes de transporte, em veículos [Sukumar e Ravi, 2016], entre outros serviços da IoT.

As Redes de Sensores Sem Fio (RSSF), tradução do inglês *Wireless Sensor Networks* (WSN) consistem de um número grande de pequenos dispositivos (sensores) com recursos computacionais limitados que atuam no sensoriamento do ambiente físico. Esses dispositivos dispõem de bateria, processador, memória e rádio para efetuar a coleta de informações do mundo real. Os sensores contidos nesta rede possuem limitação de recursos, mobilidade, cooperação e tendem a ser autônomas. As características desta rede permite a coleta de informações como temperatura, movimento, umidade, e alterações químicas no ambiente físico. As informações coletadas servem como fonte de dados para diversos serviços que beneficiam o ser humano. Esses serviços podem estar presentes em ambientes hospitalares, domiciliares, industriais, entre outros. Assim, as RSSF são fundamentais no desenvolvimento dos ambientes da IoT.

A grande maioria dos dispositivos da IoT possuem limitações de energia, processamento, armazenamento e enlaces sem fio com perdas em comunicações multi-salto. Devido as estas restrições, os padrões utilizados em outras redes não se adéquam as necessidades de comunicação da IoT. Desta forma, as organizações de padronização desenvolveram uma série de especificações sobre a comunicação de dispositivos com alcance de transmissão reduzido, limitações de energia e processamento. Assim, como ilustrado da Figura 2.3, a IoT utiliza o padrão IEEE 802.15.4 [Khanafer et al., 2014], 6LoWPAN [Sheng et al., 2013], RPL [Ishaq et al., 2013] e CoAP [Karagiannis et al., 2015].

O padrão IEEE 802.15.4 especifica a camada física e o controle de acesso para redes de área pessoal com baixas taxas de transmissão. Um padrão desenvolvido dentro do grupo de trabalho IEEE 802.15 de rede área pessoal. O padrão 802.15.4 visa o baixo custo na transmissão de dados em frequência 2.4 GHz, 915 MHz e 868 MHz entre dispositivos com no máximo 10 metros de distância. Devido às características de baixa potência e complexidade, a utilização do 802.15.4 vem crescendo em dispositivos na IoT [Sheng et al., 2013]. Além disso, organizações

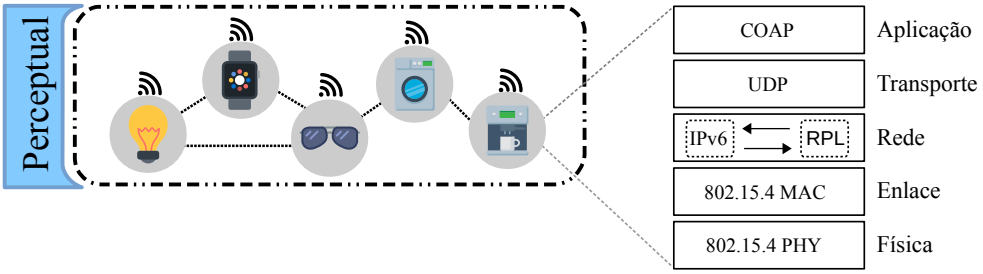


Figura 2.3: Pilha de comunicação

de normalização utilizam o padrão 802.15.4 como base no desenvolvimento de tecnologias, tais como ZigBee e WirelessHart [Gungor e Hancke, 2009].

O 6LowPAN (*IPV6 over Low Power Wireless Personal Area Network*) atua na camada de rede e descreve como aplicar o Ipv6 em camadas MAC e PHY do padrão 802.15.4. O desenvolvimento do protocolo 6LowPAN pela Força Tarefa da Internet (*Internet Task Force* - IETF) tem como principal objetivo a conectividade de redes IP (*Internet Protocol*) com redes de área pessoal com baixas taxas de transmissão da IoT. O 6LowPAN realiza uma série de fragmentações, desfragmentações e compressões nos cabeçalhos do protocolo IPv6 para proporcionar a comunicação de dispositivos de baixa capacidade de transmissão. A utilização desses processos de fragmentação e desfragmenta torna-se importante em redes na IoT, pois, o IPv6 tem um MTU máximo de 1280 octetos enquanto no IEEE 802.15.4 é de 127 octetos. Assim, o 6LowPAN cria entre o IEEE 802.15.4 e o IPv6 uma camada adaptada que mediante necessidade pode adicionar ou remover alguns cabeçalhos específicos.

O protocolo RPL (*Routing Protocol for Low-Power and Lossy*) foi projetado para efetuar o roteamento em redes de baixa potência e com perdas (LLN do inglês, *Low-Power and Lossy Networks*). O encaminhamento de informações por dispositivos da IoT ocorre com o trabalho em conjunto do protocolo RPL e o 6LoWPAN. O RPL possui suporte á tráfegos multiponto-a-ponto, ponto-a-multiponto e ponto-a-ponto. Portanto, este protocolo lida tanto com os desafios do roteamento de mensagens por dispositivos com limitações de recursos quanto com a topologia dinâmica da IoT.

Na camada de aplicação o protocolo CoAP (*Constrained Application Protocol*) permite a comunicação interativa de dispositivos com a Internet. Em seu desenvolvimento a IETF estabeleceu condições mínimas para o CoAP ser compatível com o protocolo HTTP, mas sempre mantendo o foco nas restrições energéticas e as altas taxas de falha na transmissão de pacotes nas redes da IoT [Ishaq et al., 2013]. O CoAP possui um conjunto otimizado de REST [Karagiannis et al., 2015] para M2M, com suporte a descoberta de recursos. Desta forma, este protocolo foi desenvolvido para ser um protocolo *web* genérico para redes de baixa potência e com perdas.

Esse conjunto de tecnologias permite a disseminação e encaminhamentos de informações na IoT. Desta forma, mesmo com as limitações de processamento, energia e transmissão de dados, esses protocolos permitem a troca de informações em redes de baixa potência e com perdas. A utilização dos protocolos IEEE 802.15.4, 6LoWPAN, e RPL permite o envio de dados coletados nos ambientes inteligentes da IoT. Assim, com a expectativa de ter 20 bilhões de dispositivos conectados a sua rede em 2020 [Gubbi et al., 2013], a IoT conseguirá coletar e transmitir uma enorme quantidade de dados.

2.2 Perspectiva Centrada em Dados da IoT

O objetivo principal do paradigma da IoT é o fornecimento de dados em tempo real para coisas, dispositivos computacionais, entidades inteligentes e pessoas [Aggarwal et al., 2013]. No entanto, sua heterogeneidade de dispositivos e sensores geram fluxo de dados com taxas de amostragem que variam de uma vez por dia a centenas por segundo. Além disso, essa característica divergente dificulta a coleta uniforme dos dados. Assim, desafios como coleta, processamento, gerenciamentos e interpretação são questões críticas na IoT.

2.2.1 Características dos Dados

O desenvolvimento de aplicações para ambientes inteligentes da IoT tem como ponto chave os dados coletados no mundo real. A IoT em seus ambientes possui uma grande heterogeneidade tanto nos dispositivos de sensoriamento quanto nas redes locais que transmitem os dados coletados distintos [Aggarwal et al., 2013]. Desta forma, aspectos como *geração*, *qualidade* e *interoperabilidade* dos dados se tornam importantes no desenvolvimento de aplicações nos domínios da IoT [Qin et al., 2016].

A *geração de dados* na IoT possui características como velocidade, escalabilidade, dinamicidade e heterogeneidade. A velocidade dos dados gerados pode variar de acordo com cada dispositivos, por exemplo, alguns sensores podem colher informações a uma taxa elevada de elementos por segundos e outros com taxas inferiores. Outra característica importante é a escalabilidade, pois, se espera que a extensa quantidade de coisas gere um volume grande e continuo de dados. A dinamicidade dos dados ocorre com a mobilidade, fragilidade e conectividade dos dispositivos. Por exemplo, um dispositivo pode se mover em diferentes ambientes ou passar por falhas de *hardware* ou até mesmo sofrer falhas na comunicação com outros dispositivos do ambiente. Por fim, a existência de vários coisas conectadas na IoT geram dados em diferentes formatos com vocabulários distintos.

A *qualidade dos dados* dispõe de aspectos como incerteza, redundância, ambiguidade e inconsistência. A incerteza nos dados refere-se a falta de leitura, leituras de identificadores não existentes ou falta de precisão das informações coletadas. A obtenção da redundância ocorre através da implantação de sensores de mesma características em um curto espaço do ambiente para que gerem resultados de detecção semelhantes. A presença da ambiguidade ocorre devido à existência de diferentes interpretações e necessidade dos consumidores dos dados gerados na IoT. A inconsistência dos dados refere-se a problemas como perda de pacotes durante a transmissão dos dados presentes no sensoriamento dos ambientes físicos.

A *interoperabilidade dos dados* refere-se a incompletude e semântica dos dados gerados. A mobilidade dos dispositivos na IoT pode ser um dos pontos causadores de incompletude nos dados. Por exemplo, uma determinada quantidade de dispositivos distribuídos em uma área sensoreada pode ser de grande importância para uma aplicação. Caso estes dispositivos se desloquem para outra localidade ocorrerá a incompletude dos dados necessários para uma tarefa definida nesta aplicação. A semântica dos dados refere-se ao processo de dar sentido único às informações da IoT. Assim, esse sentido único permite aos consumidores de informações da IoT entender e processar os dados.

2.2.2 Fluxo de Dados

Um fluxo de dados constitui-se de uma sequência potencialmente ilimitada de objetos de dados [Silva et al., 2013]. Esta sequência de objetos de dados pode ser gerada de forma contínua

e/ou em taxas rápidas. Os fluxos de dados possuem algumas características como método de chegada, tamanho, armazenamento e distribuição. A chegada dos objetos de dados ocorre de forma contínua ou desordenada. O tamanho do fluxo de dados pode ser potencialmente ilimitado e o armazenamento dos objetos de dados normalmente ocorre após o processamento. O processo de geração dos dados pode mudar sua distribuição probabilística ao longo do tempo. Além disso, essas características são importantes para o paradigma da IoT.

Na IoT, os fluxos de dados são gerados, em sua grande maioria, por sensores físicos distribuídos em ambientes do mundo real. A geração de dados pelos sensores ocorre em séries de tempo que podem variar de acordo com o domínio de aplicação. A taxa de amostragem pode variar de uma vez por dia a centenas por segundo. Essa variação na coleta ocorre devido a modelos heterogêneos de dispositivos, necessidades de configuração diferenciadas ou problemas no canal de transmissão do dados [Shukla e Simmhan, 2017]. A heterogeneidade dos dispositivos dificulta a coleta de uma amostra uniforme, pois, cada sensor possui características e tecnologias distintas. O sensor pode ser configurado para enviar várias informações por segundo ou uma vez por dia ou até mesmo somente transmitir dados quando existirem mudanças no valor observado. Além disso, os problemas de transmissão de dados em redes contidas na IoT também podem afetar o desempenho do encaminhamento dos dados coletados. Desta forma, pontos importantes como processamento e análise de informações dependem dos fluxos de dados.

O fluxo de dados desempenha um papel importante na internet das coisas [Qin et al., 2016]. Essa importância ocorre devido a grande quantidade de dados produzidos por todas as coisas do mundo real. Assim, o gerenciamento deste fluxo de dados se torna relevante para a IoT. Portanto, o gerenciamento do fluxo de dados possibilita uma melhora no armazenamento, processamento e análise do grande volume de dados gerados na IoT.

2.3 Considerações Finais

Este capítulo apresentou os fundamentos sobre a Internet das Coisas. Esses fundamentos mostram que o paradigma da IoT possibilita melhorias no bem-estar do ser humano em vários domínios, como, medicinal, logístico e industrial. Além disso, abordou as arquiteturas, as tecnologias contidas na rede e os padrões utilizados na comunicação entre as coisas contidas na rede da IoT. O capítulo também apresentou uma breve explicação sobre os dados da IoT. Esta explicação expôs as características dos dados e do fluxo de dados da internet das coisas, apontando as propriedades de geração, qualidade e interoperabilidade do dodos. Portanto, esse capítulo apresenta a fundamentação necessária para a compreensão da IoT e constrói um embasamento para o entendimento do próximo capítulo.

Capítulo 3

Gerenciamento de Fluxo de Dados na IoT

O gerenciamento de fluxos de dados na IoT torna-se cada vez mais desafiador com o grande volume de dados gerados. O aumento no volume e na velocidade dos dados gerados pode consumir os recursos de comunicação da rede e piorar requisitos como latência. Além disso, os tipos de dados gerados pela IoT, por exemplo, fluxos de vídeo e medições em tempo real, aumentam o estresse sobre a infraestrutura de rede. Desta forma, técnicas como filtragem e processamento na borda da rede, minimizam a carga de tráfego injetada na infraestrutura de comunicação são cruciais nos ambientes da IoT.

As técnicas de gerenciamento de fluxo de dados na IoT vem sendo desenvolvidas com base em abordagens, por exemplo, baseadas em Computação em Nuvem (CC), Computação em Névoa (Fog) e Rede Definida por Software (SDN). A CC (*Cloud Computing*) permite a utilização de recursos computacionais e de armazenamento de servidores compartilhados via internet. A Fog (*Fog Computing*) disponibiliza recursos de *hardware* e *software* mais próximos do utilizador final. A SDN (*Software Defined Networking*) propõe uma arquitetura dinâmica e gerenciável para o controle de fluxo da rede.

O restante do capítulo apresenta os fundamentos necessários à compreensão do contexto da pesquisa, do problema e da proposta. A Seção 3.1 apresenta os conceitos de computação em Nuvem, bem como as vantagens e desvantagens na integração da Nuvem com a IoT. A Seção 3.2 apresenta uma visão geral da computação em Névoa em conjunto com a IoT. A Seção 3.3 consiste de uma breve explicação sobre os conceitos das Redes Definidas por *Software*. Na Seção 3.4 apresenta trabalhos contidos na literatura que utilizam as abordagens descritas nas seções anteriores. A Seção 2.3 mostra um resumo do conteúdo abordado no capítulo.

3.1 Computação em Nuvem

O rápido avanço das tecnologias de processamento, armazenamento, comunicação e o sucesso da internet, contribuíram para o desenvolvimento do modelo operacional denominado Computação em Nuvem (CC - *Cloud Computing*). A CC não se trata de uma tecnologia, pois, é vista como um modelo operacional de negócios originados de tecnologias já existentes como, virtualização e Web [Bilal et al., 2014]. Seu modelo operacional fornece, por meio da internet, acesso ubíquo e sob demanda a um conjunto de recursos computacionais elásticos e configuráveis.

Na literatura existe uma ampla quantidade de trabalhos que definem a CC. [Zhang et al., 2010] define a CC como um ambiente de capacidade altamente elástica que disponibiliza, via internet, recursos computacionais (*hardware* e *software*) sob demanda para o usuário final. Na visão de [Mell e Grance, 2011] a CC é um modelo conveniente e sob demanda para um *pool* compartilhado de recursos computacionais configuráveis que podem

ser rapidamente provisionados e lançados com um mínimo de esforço no gerenciamento. Para [Rittinghouse e Ransome, 2009] a computação em nuvem é um sistema paralelo e distribuído composto por um *pool* compartilhado de recursos virtualizados (por exemplo, rede, servidor, armazenamento, aplicativo e serviço) em centros de dados em larga escala. Esses recursos podem ser provisionados, reconfigurados e explorados dinamicamente por um modelo econômico *pay-per-use* (pagamento por uso) no qual o consumidor paga pela quantidade de recursos utilizados.

O conceito da CC concentra os recursos de processamento e armazenado em grandes centros de dados (DC - *Data Center*). Esses centros são compostos por máquinas de alto desempenho que estão interligadas por conexões com alta largura de banda. Os recursos disponíveis são concedidos aos usuários por meio da internet. Desta forma, independente da plataforma utilizada, os usuários executam solicitações à nuvem e recebem os dados processados sem a necessidade de adquirir máquinas de alto desempenho [Vacca, 2017]. Os autores em [Mell e Grance, 2011] definem outras características essenciais na CC:

Recursos sob demanda: o usuário pode acrescentar ou reduzir os recursos computacionais consumidos de acordo com a necessidade existente no momento e sem precisar ter interação humana com o provedor dos recursos.

Agrupamento de recursos: o provedor visa disponibilizar recursos para múltiplos clientes via modelo arquitetônico *multi-tenant*[1]. Esses recursos são alocados dinamicamente de acordo com a demanda de consumo e sua localização é transparente aos clientes. Desta forma, o cliente geralmente não tem controle ou conhecimento sobre a localização exata dos recursos oferecidos.

Serviços mensuráveis: os sistemas em CC coordenam e otimizam automaticamente o uso de recursos computacionais fomentando a capacidade de medição em diferentes níveis de granularidade. Desta forma, o monitoramento e controle dos recursos proporcionam uma transparência tanto para o provedor quanto para o consumidor dos serviços oferecidos.

Elasticidade rápida: os recursos computacionais podem ser providos e concedidos elasticamente, ou seja, a escalabilidade de recursos segue em conformidade com a demanda. Desta forma, a disponibilidade de recursos parece ilimitada para o consumidor. Assim, o mesmo pode adquirir recursos computacionais em qualquer quantidade e a qualquer momento.

A computação em nuvem possui uma série de implementações baseadas em serviços. Esses serviços podem variar desde o provisionamento de aplicações até a recursos de infraestrutura em rede. Os principais modelos de serviços em CC podem ser classificados em Infraestrutura como Serviço (*IaaS*), Plataforma como Serviço (*PaaS*) e Software como Serviço (*SaaS*) [Armbrust et al., 2010, Vacca, 2017]. O modelo de serviço *IaaS* (*Infrastructure as a Service*) fornece ao cliente, recursos de infraestrutura sob demanda, ou seja, paga somente pelos recursos utilizados. Esse modelo permite ao cliente contratar servidores virtuais e outros dispositivos de infraestrutura de acordo com a necessidade exigida no momento. Por exemplo, uma empresa poderia contratar servidores de acordo com a demanda de acesso ao seu *site* ao invés de investir um capital alto em equipamentos físicos. Além disso, como mostra a Figura 3.1(b), o cliente possui plena autonomia na implantação de sistemas operacionais e aplicativos, em geral. Já, o provedor fica responsável por gerenciar e controlar a infraestrutura como, armazenamento, método de virtualização e componentes de redes fornecidos em nuvem.

[1]modelo arquitetônico que permite múltiplos inquilinos (empresas ou clientes) compartilharem os mesmos recursos físicos contidos em ambientes da CC.

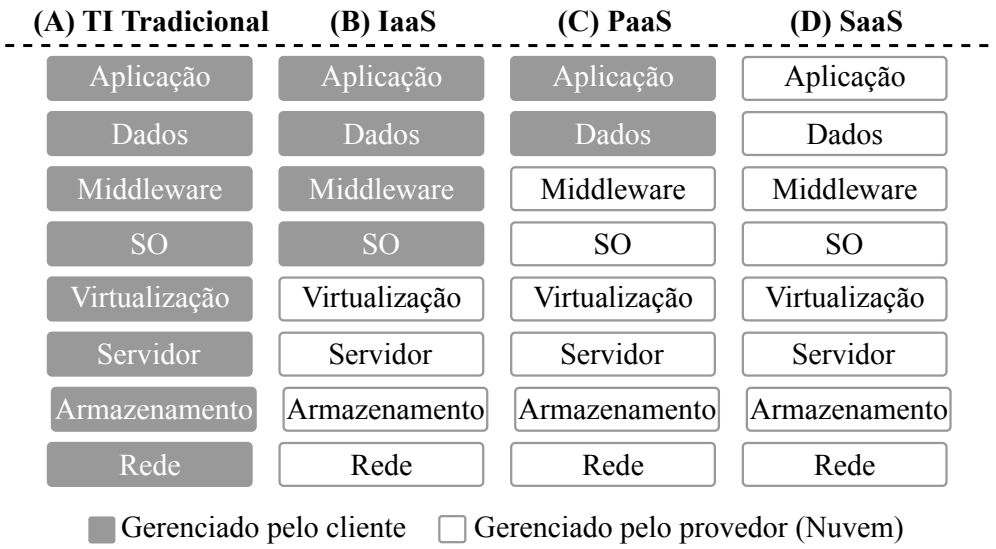


Figura 3.1: Comparação entre modelos de serviços, adaptado de [Vacca, 2017].

O modelo de serviço *PaaS* (*Platform as a Service*) disponibiliza um ambiente para o desenvolvimento de aplicações simples e sofisticadas para empresas. Esse modelo provê um ambiente de apoio ao ciclo de construção de aplicações, por exemplo, fornecimento de suporte ao levantamento de casos de uso, codificação, teste até a manutenção. Além disso, como mostra a Figura 3.1(c), o provedor abstrai as complexidades de implantação e gerenciamento de requisitos de *hardware* e *software* como, rede, sistema operacional, *middleware* e outros recursos. Desta forma, o cliente reduz o custo de implantação, gerencia somente as aplicações que desenvolve e o provedor de serviços em nuvem efetua o gerenciamento do restante da estrutura.

O modelo de serviço *SaaS* (*Software as a Service*) provê sob demanda uma solução de *software* completa para o cliente. Neste modelo, como apresenta a Figura 3.1(d), o provedor de serviços é responsável por gerenciar toda a infraestrutura de *hardware* (por exemplo, rede, servidor e armazenamento) e de *software* (por exemplo, sistemas operacionais e aplicações), abstraindo toda a infraestrutura da nuvem para o cliente. Desta forma, o gerenciamento efetuado pelo cliente se restringe a definições de configuração da aplicação. Assim, o mesmo utiliza as aplicações fornecidas pelo provedor de serviços sem se preocupar com o gerenciamento ou controle da infraestrutura existente na CC.

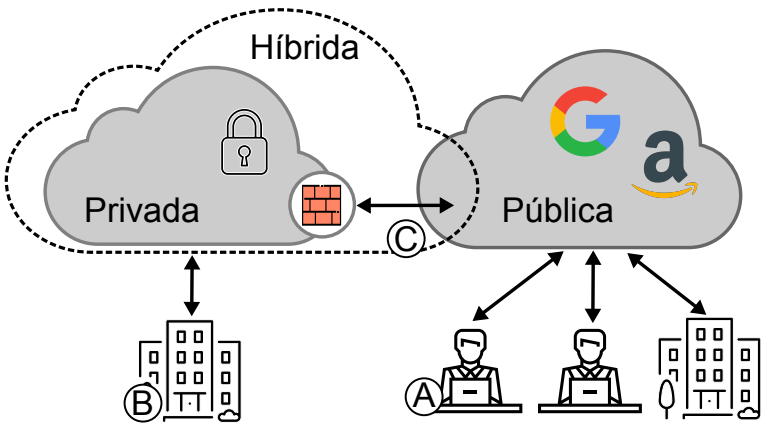


Figura 3.2: Modelos de implantação

Os modelos de serviços providos pela CC são implementados de acordo com suas propriedades. Elas podem variar desde o tamanho dos recursos existente na nuvem até as restrições de acesso de cada cliente. Desta forma, como apresenta a Figura 3.2, existem três modelos principais de implementação: nuvem pública, privada e híbrida [Mell e Grance, 2011].

Nuvem pública: são ambientes desenvolvidos e comercializadas por terceiros. Este terceiro implementa uma infraestrutura que disponibiliza comercialmente um leque de recursos computacionais para o público em geral. Como mostra a Figura 3.2(a), nesse ambiente várias organizações ou indivíduos compartilham, através de virtualização, recursos computacionais disponíveis em nuvem. Empresas como Google [Google, 2017] e Amazon [Amazon, 2017] são exemplos de provedores de nuvem pública.

Nuvem privada: são ambientes implementados de forma centralizada em uma organização. Este ambiente centraliza os recursos computacionais da organização e os oferecem aos seus próprios departamentos. Desta forma, os departamentos da organização irão dispor de recursos elásticos, sob demanda e escalável. Como mostra a Figura 3.2(b), a organização assume ao mesmo tempo, o papel de provedor e consumidor de nuvem. Assim, ela possui total controle na administração de recursos e nas garantias de segurança dos seus dados.

Nuvem híbrida: combinação entre os ambientes da nuvem pública e privada. Apresentado na Figura 3.2(c), na nuvem híbrida os ambientes públicos e privados atuam de forma independente e se conectam quando necessário. A conexão entre os dois ambientes permite somar vantagens existentes em ambas como, a segurança da nuvem privada e a maior elasticidade da nuvem pública. Por exemplo, seria possível tratam as informações sensíveis na nuvem privada que não possui compartilhamento com outras empresas e armazenar dados não críticos na nuvem pública que possui menor custo em recursos.

3.1.1 Integração Nuvem e IoT

As limitações de recursos na IoT dificultam o desenvolvimento de serviços e aplicações em sua rede. Os objetos contidos na IoT, em sua grande maioria dispõe de *hardware* restrito com pouco poder de processamento, armazenamento e energia [Gubbi et al., 2013]. Essa inferioridade em conjunto com a heterogeneidade de *hardware* agrega complexidades no desenvolvimento de serviços para a IoT. Por este motivo, a integração entre IoT e Nuvem minimizam as dificuldades referentes à comunicação, computação e armazenamento [Botta et al., 2016]. Na comunicação, a Nuvem oferece uma solução eficaz e barata para conectar, controlar, e gerenciar qualquer coisa, em qualquer lugar a qualquer momento. Na computação, os dados coletados na IoT são agregados e processados em nós com recursos computacionais ilimitados. Por fim, no armazenamento, a nuvem oferece uma estrutura de armazenamento em larga escala e de longa duração para a grande quantidade de dados gerados na IoT.

A Figura 3.3 apresenta a integração entre ambientes da IoT e a CC. Na parte inferior da figura contém três ambientes da IoT: domiciliar, transporte e hospitalar. Esses ambientes são compostos por objetos como, relógio, lâmpadas, veículos, sensores de temperatura e velocidade que utilizam a infraestrutura da Internet para enviar os dados coletados até a nuvem. Na nuvem os dados são armazenados e processados sem preocupações com a disponibilidade de recursos computacionais. Desta forma, a CC minimiza esforços no desenvolvimento de serviços e aplicações baseadas no paradigma da IoT.

A integração entre IoT e CC contribui para o surgimento de novos de conceitos baseados em nuvem para a IoT. Por exemplo, o conceito *SaaS*, *SenaaS*, *DaaS* e *TaaS* [Botta et al., 2016].

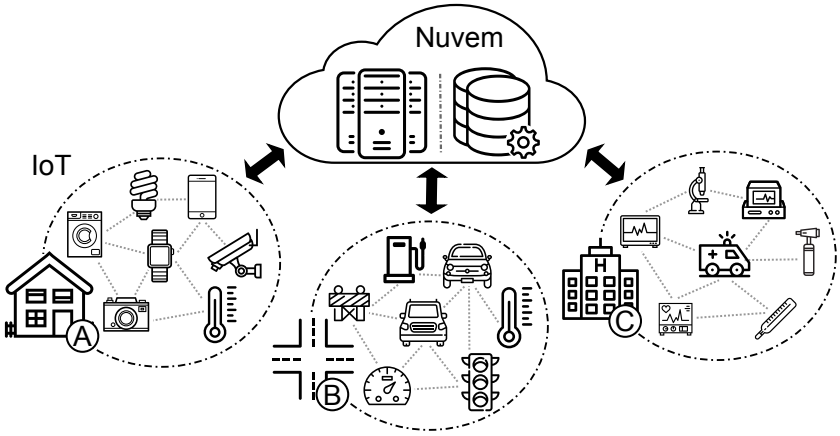


Figura 3.3: Integração IoT e CC

O *SaaS* (*Sensing-as-a-Service*) provê acesso ubíquo aos dados de sensores. O *SenaaS* (*Sensor-as-a-Service*) proporciona o gerenciamento remoto e onipresente dos sensores. O *DaaS* (*Data-as-a-Service*) possibilita o acesso ubíquo a qualquer dado coletado. Já, a *TaaS* (*Things-as-a-Service*) permite agregar e abstrair recursos heterogêneos de acordo com a semântica dos objetos. Assim, a integração entre IoT e Nuvem favorece a criação de novos projetos que exploram a grade volume de dados da IoT.

O desenvolvimento de projetos para a IoT em conjunto com a CC cresce com o amadurecimento da Internet das Coisas. Na literatura existe uma quantidade ampla de projetos que desenvolvem plataformas de *middleware* baseadas em Nuvem para a IoT [Mineraud et al., 2016]. Essas plataformas facilitam o desenvolvimento de aplicações voltadas à IoT, por exemplo, os projetos Fiware [Fiware, 2017b] e OpenIoT [OpenIoT, 2017] fornecem plataformas que gerenciam tanto os dados gerados quanto os objetos contidos em ambientes da IoT. Além disso, esses projetos baseados em Nuvem minimizam as complexidades da IoT. Desta forma, a integração entre esses paradigmas colaboram para o desenvolvimento da IoT.

A computação em nuvem fornece vários benefícios para IoT, dentre eles, a elasticidade de recursos computacionais [Yi et al., 2015]. No entanto, a IoT precisa enviar os dados coletados até a nuvem para usufruir de seus recursos computacionais. Contudo, o volume do tráfego de dados da IoT para a nuvem pode causar um consumo excessivo dos recursos da infraestrutura da internet, visto que a IoT gera um grande volume de dados [Gubbi et al., 2013]. Além disso, na IoT existem serviços que demandam de baixa latência, ou seja, serviços sensíveis ao atraso como, operações de gerenciamento de emergências [Yang et al., 2013]. Desta forma, a IoT necessita de modelos que ofereçam recursos computacionais com e baixa latência no acesso às informações.

3.2 Computação em Névoa

Uma perspectiva desenvolvida pela empresa *Cisco Systems*[2], a computação em Névoa (*Fog*) se resume em uma plataforma altamente virtualizada que fornece recursos computacionais entre os dispositivos finais e os provedores tradicionais de serviços em Nuvem [Bonomi et al., 2014]. Esse conceito fornece serviços de processamento, armazenamento e comunicação na borda da rede sem depender diretamente dos servidores tradicionais da Nuvem. Assim, a disponibilidade de recursos próximo da borda e reduz os problemas de latência na rede.

[2] `http://www.cisco.com/`

A *Fog* agrega várias vantagens no desenvolvimento de serviços e utilização de recursos da rede. Um dos principais benefícios é a baixa latência no atendimento de requisições, tornando-a ideal para serviços de voz sobre IP, jogos *online*, etc., além disso, o fato das requisições serem processadas próximo ao usuário final, podem contribuir para a descentralização dos serviços e na minimização da sobrecarga na Nuvem e no núcleo da rede. No entanto, por ser constituída de dispositivos de recursos limitados e não dedicados, a *Fog* possui limitações no fornecimento de recursos computacionais e complexidades no desenvolvimento de aplicações [Mahmud e Buyya, 2016].

As complexidades existentes na computação em Névoa implicam em uma série de características que a tornam uma extensão não-trivial da Nuvem. Essas características são listadas em [Bonomi et al., 2012]:

Reconhecimento de localidade e baixa latência: a origem da computação em Névoa pode ser atribuída às propostas anteriores envolvendo pontos de acesso com suporte a serviços sofisticados na borda da rede, incluindo aplicações com requisitos de baixa latência.

Distribuição geográfica: em nítido contraste com a computação mais centralizada da Nuvem, os serviços e aplicações orientadas para Névoa demandam implantações amplamente distribuídas.

Suporte a redes de sensores em larga escala: redes de sensores para monitoramento do meio ambiente e o *Smart Grid* são outros exemplos de sistemas inerentemente distribuídos, que requerem recursos de computação e armazenamento distribuídos.

Grande número de nós: é uma consequência da ampla distribuição geográfica como evidenciada pelas tecnologias de redes de sensores e pela aplicação *Smart Grid* em particular.

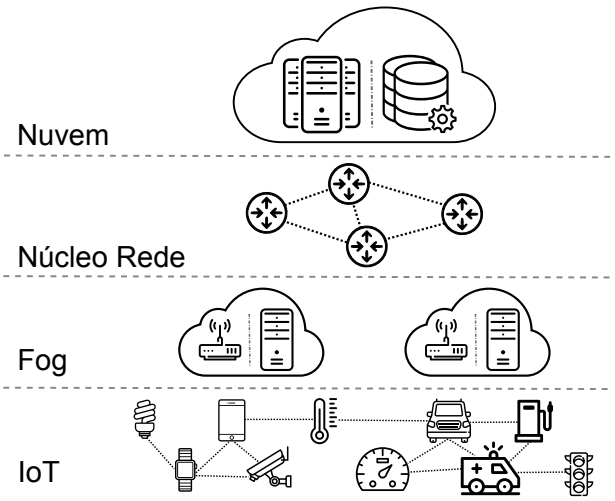


Figura 3.4: Arquitetura Fog, adaptado de [Bonomi et al., 2012]

Suporte a computação móvel: em muitas aplicações de Névoa é essencial uma comunicação direta com os dispositivos associados para permitir o suporte às diferentes técnicas que podem ser empregadas na computação móvel.

Interações em tempo real: as principais aplicações em *Fog* envolvem interações em tempo real, em vez do processamento em lote. Como a *Fog* está localizada longe dos principais centros de processamento em Nuvem, se faz necessário a disponibilização e o gerenciamento dos recursos locais para auxiliar o cumprimento dos requisitos de tempo de execução das aplicações.

Predominância do acesso sem fio: para os dispositivos IoT algum protocolo e comunicação sem fio como RFID [Jia et al., 2012], *Bluetooth* [Chang, 2014], *ZigBee* [Gungor e Hancke, 2009], Wi-Fi ou LTE [Cao et al., 2014] é a única forma possível de conexão em rede.

Heterogeneidade: as Nuvens são geralmente ambientes fechados, que utilizam componentes de *hardware* de um mesmo fornecedor ou ambientes e linguagens de programação proprietárias. No entanto, os dispositivos da *Fog* podem ser oferecidos por diversos fabricantes, empregar diferentes ambientes e envolver vários linguagens e protocolos.

Interoperabilidade e federação: o suporte contínuo e integrado de certos serviços requerem a cooperação de diferentes provedores.

Análise de dados em tempo real: interagindo com a Nuvem e perto das fontes de dados, a *Fog* esta bem localizada para desempenhar um papel significativo no processamento de dados com restrições de tempo real.

A Figura 3.4 apresenta a arquitetura e ilustra uma implementação da computação em Névoa. A camada inferior ilustra os milhares de dispositivos conectados que se comunicam entre si em tempo real. A próxima camada, representa os dispositivos que compõem a infraestrutura para implantação da Fog. Esses dispositivos proveem conectividade para a camada inferior através de protocolos como, *Bluetooth* [Chang, 2014], *ZigBee* [Gungor e Hancke, 2009], Wi-Fi ou LTE. Além disso, esta camada fornece a conectividade com os serviços disponíveis nos provedores de Nuvem. A camada superior ilustra os recursos providos pela Nuvem, onde são armazenados e processados dados que não requerem consumo em tempo real. Desta forna, a computação em Névoa distribui e coloca os dados mais próximos do usuário final, minimiza a latência dos serviços, melhora a qualidade dos serviços e remove outros possíveis obstáculos relacionados com a transferência de dados.

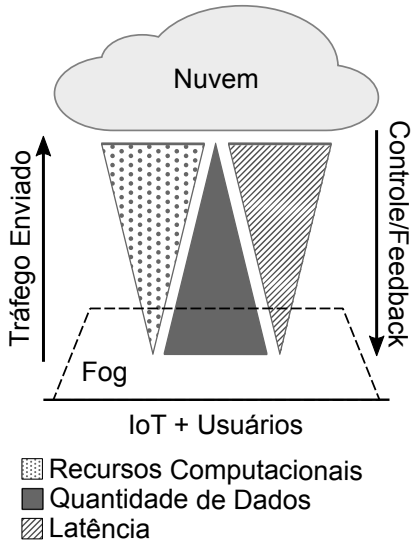


Figura 3.5: Comparação Fog e CC

A computação em Névoa não é um conceito que substitui totalmente a utilização da computação em Nuvem. Na Figura 3.5 pode ser visto uma comparação na utilização da Fog e da Nuvem na IoT. A Nuvem possui uma grande quantidade de recursos computacionais. Contudo, o tráfego de dados brutos causam um consumo excessivo de recursos da rede para atingir seus servidores. Além disso, agrega um aumento de latência aos serviços oferecidos na IoT. A Fog oferece recursos mais próximos da borda e minimiza a latência em serviços da IoT, no entanto, a disponibilidade dos recursos computacionais é limitada. Logo, tanto Fog quanto Nuvem agregam benefícios ao desenvolvimento de serviços na IoT.

3.3 Rede Definida por Software

As redes definidas por software (SDN - *Software Defined Networking*) surgiram com o objetivo de simplificar o gerenciamento das redes atuais. No modelo atual, os elementos de rede são dependentes de APIs proprietárias que inviabilizam a implementação de novos protocolos e arquiteturas. Por este motivo, a ideia de redes programáveis surgiram com o objetivo de separar a camada de infraestrutura da camada de controle, facilitando o desenvolvimento de novas soluções que atendam as limitações (endereçamento, mobilidade, etc) das redes atuais [Nunes et al., 2014].

O conceito SDN possibilita a dissociação do *hardware* de encaminhamento (plano de dados) das decisões de controle (plano de controle) da rede [Farhady et al., 2015]. Com este design, o controle de rede pode ser feito separadamente no plano de controle sem afetar os fluxos de dados. Como tal, a inteligência de rede pode ser retirada dos dispositivos de comutação e colocada em controladores. Além disso, os dispositivos de comutação agora podem ser controlados externamente por *software*. O desacoplamento do plano de controle do plano de dados oferece não apenas um ambiente programável mais simples, mas também uma maior liberdade para o *software* externo definir o comportamento de uma rede.

Apresentada na Figura 3.6, a arquitetura da SDN está dividida em três camadas: infraestrutura, controle e as aplicação. A camada de infraestrutura consiste em elementos de rede e dispositivos que realizam a troca e o encaminhamento de pacotes (dados). A camada de controle provê a funcionalidade de controle que supervisiona os encaminhamentos dados através

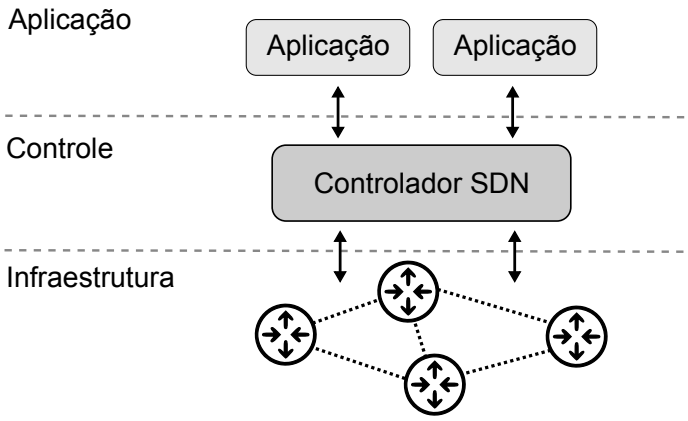


Figura 3.6: Arquitetura SDN

de uma *interface* aberta. A camada de aplicação engloba as aplicações desenvolvidas através de API's para usuários finais que utilizam os serviços de comunicação da SDN.

A utilização dos conceitos de SDN e IoT em um mesmo contexto flexibiliza o gerenciamento das redes da Internet das Coisas. Os ambientes da IoT possuem características que impedem o uso de protocolos e mecanismos comumente utilizados em redes de computadores tradicionais. A integração com SDN viabiliza a utilização de *software* e linguagens de alto nível para programar as redes da IoT. Desta forma, a SDN permite lidar com as especificações dos dispositivos, cenários e aplicações existentes na IoT de forma ágil e flexível.

3.4 Revisão Literária

Nesta seção serão apresentados os trabalhos selecionados na literatura sobre gerenciamento de dados da IoT. A revisão foi dividida em três abordagens, sendo elas, Rede Definida por Software (SDN), Computação em Névoa (Fog) e Computação em Nuvem (CC). A seguir, teremos o detalhamento e o apontamento sobre cada trabalho selecionado.

3.4.1 Abordagens Baseadas em SDN

Os benefícios de empregar técnicas baseadas em SDN nos ambientes da IoT estão se tornando reconhecidos em múltiplos domínios além do transporte inteligente, que são discutidos principalmente por pesquisadores e profissionais da indústria. No trabalho [Wu et al., 2015] é proposto um sistema para controle de fluxo de dados baseado em SDN, denominado *UbiFlow*. O sistema possui um controlador que diferencia a programação do fluxo baseado na exigência por dispositivos. Além disso, o *UbiFlow* apresenta uma visão do estado da rede e otimiza os pedidos de fluxo da IoT. Na avaliação, os autores comparam o *UbiFlow* com os sistemas *DevoFlow* e *Hedera*, onde mostrou que seu sistema possui resultados melhores que ambos. Semelhante ao [Wu et al., 2015], o artigo [Qin et al., 2014] busca desenvolver uma abordagem dinâmica com níveis de qualidade diferenciados para as distintas tarefas na IoT. O projeto, denominado MINA (*Multinetwork INformation Architecture*), visa criar um *middleware* com um controlador em diferentes camadas da IoT. Este *middleware* permite suporta comandos para diferentes fluxos e explora eficientemente os recursos de características multi-rede da Internet das Coisas.

Em [Nastic et al., 2015], os autores propõem um *framework* para sistema definido por *software* em nuvem para IoT. Este *framework*, denominada *SDG-Pro*, permite o desenvolvimento de um *software* definido por *gateway* com abstração de programação. Além disso, seus pontos

de dados e controle possibilitam que desenvolvedores possam interagir com os fluxos de dados sensoreados. Essa interação torna possível, funcionalidades que unificam os fluxos de dados, independente da comunicação. No trabalho [Jararweh et al., 2015] são apresentados princípios como, rede definida por *software* (SDN), armazenamento definido por *software* (SDStore) e segurança definida por *software* (SDSec). Os autores utilizaram esses princípios para desenvolver um modelo de arquitetura para IoT, denominada IoT definida por *software* (SDIoT). Esse modelo permite acelerar e facilitar operações de controle e gerenciamento na IoT.

3.4.2 Abordagens Baseadas em Fog

Na literatura existem alguns trabalhos que propõem soluções baseadas em Fog para IoT. O trabalho de [Gazis et al., 2015] propõem uma solução industrial no contexto de aplicações de manutenção preventiva. Sua solução, denominada AOP (*Adaptive Operations Platform*), utiliza a tecnologia proprietária DMo[3] para manipular as regras de tráfego da rede. Esta plataforma permite aos fabricantes de equipamentos, controle supervisionado e aquisição de dados de sua infraestrutura. O artigo [Aazam e Huh, 2014] apresenta um *gateway*[4] para minimizar os gastos excessivos de recursos da rede e da nuvem. Ele propõe o *Smart Gateway* um intermediário que faz um pré processamento dos dados antes de os enviar para a nuvem. Seu principal objetivo é evitar comunicações desnecessárias com a nuvem. Semelhante ao [Aazam e Huh, 2014], o trabalho de [Rahmani et al., 2015] apresenta um *smart gateway*, denominado *UTGATE*. Ele lida com desafios em sistemas de saúde e explora meios para controlar os sensores distribuídos em um ambiente *e-Health*[5]. Após a avaliação, os autores provaram que o *gateway* proposto permitiu uma maior eficiência energética dos sensores contidos na rede.

O trabalho de [Prazeres e Serrano, 2016] introduz o paradigma FoT(*Fog of Things*) e propõem o desenvolvimento de uma plataforma de auto-organização, denominada SOFT-IoT (*Self-Organizing Fog of Things*). A plataforma fornece um *middleware* orientado a serviços que possui uma *interface* de acesso aos dispositivos via serviço *web* RESTful [Karagiannis et al., 2015]. Além disso, sua plataforma utiliza um corretor MQTT [Karagiannis et al., 2015] que possibilita a troca de mensagens entre o *gateway* e os dispositivos através de *driver* MQTT integrado com o serviço *web*. Diferente de [Prazeres e Serrano, 2016], o artigo [Cheng et al., 2015] busca desenvolver uma plataforma de análise de dados, denominada *GeeLytics*. Os autores afirmam que esta plataforma pode executar o processamento em tempo real tanto na borda quanto na nuvem de forma dinâmica e transparente. No entanto, o documento apenas discute a motivação e o projeto preliminar de sua arquitetura.

3.4.3 Abordagens Baseadas em CC

Na literatura existem projetos que visam oferecer estruturas inteligentes que permitem o desenvolvimento de aplicação para IoT em conjunto com a computação em nuvem [Mineraud et al., 2016]. Esses projetos podem ser desenvolvidos como um produto proprietário, por exemplo, Arkessa [Arkessa, 2017], Carriots [Carriots, 2017] ou Niagara [Niagara, 2017], mas também existem iniciativas de código aberto (*Open Sourse*) como, OpenIoT [OpenIoT, 2017], Fiware [Fiware, 2017b], Sofia [Sofia, 2016b], Nimbits [Nimbits, 2016],

[3] Tecnologia de software DMo (*Data in Motion*) fornece gerenciamento de dados e análise de primeira ordem na borda da rede. Disponível em <https://developer.cisco.com/site/data-in-motion/>

[4] Intermediário destinado a interligar redes, separar domínios, ou até mesmo traduzir protocolos.

[5] Ambiente inteligente da área de saúde.

LinkSmart [LinkSmart, 2016] e ThingSpeak [ThingSpeak, 2016b]. No entanto, este trabalha busca descrever as características somente das plataformas de código aberto.

Projeto OpenIoT

O projeto OpenIoT *(Open Source Cloud solution for the Internet of Things*) oferece uma plataforma de *middleware* de código aberto totalmente descentralizada [Soldatos et al., 2012]. Uma plataforma desenvolvida no projeto FP7-287305 com co-financiamento da Comissão Europeia, o OpenIoT provê uma infraestrutura versátil que fornece conectividade com dispositivos de capacidades restritas, tais como, sensor e RFID. Esta conectividade permite que a plataforma disponibilize funcionalidades para a filtragem e seleção dinâmica dos fluxos de dados gerados por estes dispositivos. O OpenIoT oferece também uma quantidade ampla de ferramentas de visualização para o desenvolvimento de aplicações na IoT. Assim, a plataforma de *middleware* do OpenIoT permite o desenvolvimento de aplicações para IoT baseadas em nuvem com um gasto mínimo de codificação.

A plataforma OpenIoT possui uma arquitetura dividida em elementos, sendo eles, *middleware*, armazenamento na nuvem, processo de agendamento, serviços de entrega e componente de solicitação, apresentação e monitoramento [Soldatos et al., 2015]. O *middleware* do OpenIoT utiliza uma extensão das bibliotecas de redes mundiais de sensores, denominado X-GSN (*Extended Global Sensor Networks*) [Calbimonte et al., 2014], como meio para a implementação da *interface*, filtros e agregação de dados provenientes dos dispositivos e objetos conectados na IoT. O armazenamento do fluxo de dados na nuvem ocorre por meio do LSM (*Linked Stream Middleware*), que transforma os dados provenientes dos sensores em dados vinculados semanticamente em formato RDF (*Resource Description Format*). O elemento de agendamento efetua o processamento de solicitações para implementações de serviços sob demanda e garante o acesso apropriado aos recursos. O serviço de entrega busca através de consultas expressas em linguagem semântica os dados solicitados por serviços implementados na plataforma. O componente de solicitação, apresentação e monitoramento disponibiliza um conjunto de ferramentas para especificar, formular, apresentar e monitorar através de *interface* gráfica (GUI do inglês, *Graphical User Interface*) as solicitações de dados feitos por um serviço. Assim, esta arquitetura segmentada possibilita a implementação de serviços da IoT sobre a plataforma OpenIoT.

A Figura 3.7 mostra o processo de implementação de serviços da IoT sobre a plataforma OpenIoT [Soldatos et al., 2012]. A primeira etapa resume-se na definição dos serviços. Agora ocorre a definição do serviço de solicitação e a identificação e seleção dos objetos. A etapa seguinte permite a formulação e a apresentação do serviço no ambiente em nuvem. Nesta fase realiza-se a composição do serviço de acordo com o contexto da aplicação, a localização, a restrição e se o serviço atua em tempo real. Na terceira etapa ocorre a configuração dos objetos envolvidos na prestação do serviço, onde se efetua a parametrização dos parâmetros de medição utilizados no compartilhamento de recurso. O desenvolvimento de aplicações na plataforma ocorre com um gasto mínimo com codificação, pois, todo o processo de implantação dá-se por *interface* gráfica.

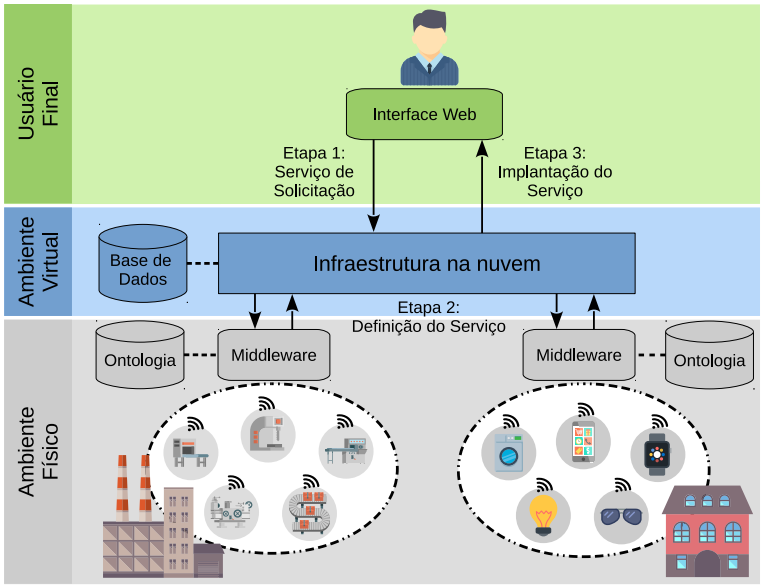


Figura 3.7: Funcinamento OpenIoT, adaptado de [Soldatos et al., 2012].

A plataforma OpenIoT permite a convergência da IoT com a computação em nuvem. Esta integração possibilita a incorporação do grade volume de dados e aplicações da IoT com a infraestrutura elástica da computação em nuvem. Além disso, sua plataforma proporciona a implantação de aplicações semanticamente interoperáveis e o controle de parâmetros associados a qualidade de serviço. Desta forma, suas funcionalidades fornecem uma base para o desenvolvimento de novas aplicações nos domínios da IoT.

Projeto Fiware

O projeto Fiware oferece uma infraestrutura baseada em nuvem para criação e entrega de serviços com um baixo custo de implantação. A plataforma Fiware fornece um conjunto simples de APIs (*Application Programming Interfaces*) que facilitam o desenvolvimento de aplicações inteligentes. Essas APIs possuem compatibilidade com OCCI (*Open Cloud Computing Interface*) [Edmonds et al., 2015]. Além disso, o Fiware baseia-se no projeto OpenStack [Sefraoui et al., 2012] uma plataforma de computação em nuvem de código aberto onipresente para nuvens públicas e privadas. Assim, sua referência de código aberto permite aos desenvolvedores, provedores de serviços, empresas e outras organizações desenvolverem produtos inovadores com base em tecnologias Fiware.

A plataforma Fiware possui uma infraestrutura dividida em elementos funcionais, denominado habilitador genérico (GE do inglês, *Generic Enablers*) [Palo et al., 2013]. A principal plataforma fornecida pelo projeto Fiware dispõe de GE's como, Estrutura de Entrega de Serviço, Hospedagem na Nuvem, Gerenciamento de Dados, Segurança, Interface com a Rede e Dispositivos (I2ND) e IoT. A Estrutura de Entrega de Serviço permite criar, publicar, gerenciar e consumir serviços da internet do futuro (FI do inglês, *Future Internet*). A Hospedagem na Nuvem fornece recursos para processamento, armazenamento e rede para os serviços fornecidos. O Gerenciamento de dados fornece facilidades como, acesso eficiente, processamento e análise de dados para gerar conhecimento a partir de um grande volume de dados. A Segurança disponibiliza mecanismos que assegurem os requisitos de confidencialidade e privacidade dos serviços utilizados. A I2ND supre a necessidade de conectividade dos serviços prestados. Para a IoT, a plataforma dispõe de habilitadores genéricos que permitem a utilização de serviços em dispositivos com recursos limitados e heterogêneos.

O Fiware IoT fornece GE's para objetos contidos na rede tornem-se disponíveis, pesquisáveis, acessíveis e utilizáveis no desenvolvimento de aplicações. O Fiware IoT expõem os dispositivos e recursos para os desenvolvedores como entidades de contexto NGSI (*Next Generation Services Interface*) [Moltchanov e Rocha, 2014]. Desta forma, os desenvolvedores se atentam somente na leitura de atributos de entidades para agregar as informações de sensores. Além disso, os desenvolvedores não precisam lidar com a complexidade e a grande heterogeneidade das tecnologias existente nos cenários IoT.

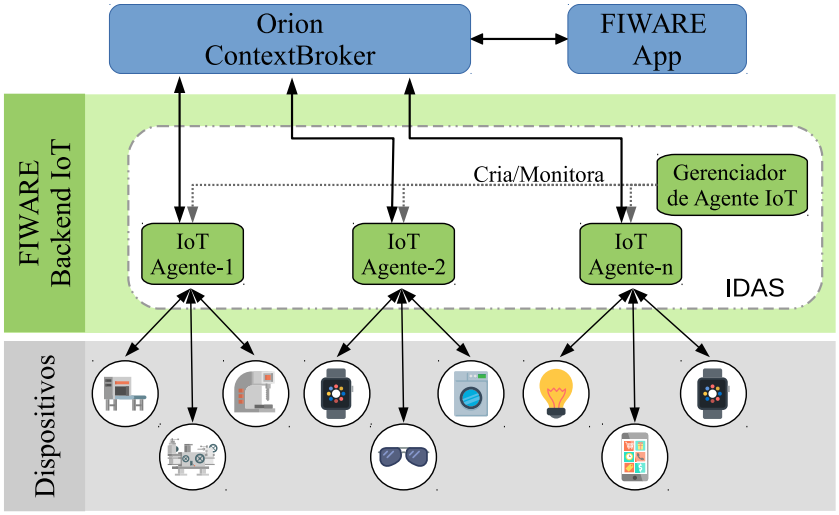


Figura 3.8: Arquitetura Fiware IoT, adaptado de [Fiware, 2016]

A Figura 3.8 apresenta um cenário simplificado do funcionamento do FIWARE IoT [Fiware, 2016]. Este cenário simples considera somente a utilização da GE IDAS e Orion. O componente IDAS gerencia e íntegra os dispositivos conectados ao ambiente. Ele utiliza uma arquitetura modular, denominada de Agente IoT, para suporta os vários protocolos existentes na rede. Desta forma, antes de selecionar o agente IoT, os integrantes da rede precisam determinar o protocolo que será utilizado na comunicação. O elemento *Orien ContextBroker* faz a tradução dos recursos da IoT para entidades de contexto NGSI [Fiware, 2017a]. Assim, os desenvolvedores acessam dados da IoT como atributos de entidades representativas e podem enviar comandos através de atualização atributos relacionados a dispositivos. Portanto, o Fiware IoT dispõe de funcionalidades robustas com baixo grau de complexidade no desenvolvimento e implementação.

O projeto Fiware oferece soluções com funcionalidades robustas e altamente escalável. Sua plataforma permite conectar aplicações e serviços de forma inteligente e aproveitar todo o potencial dos recursos oferecidos pela rede. Além disso, o projeto pode potencializar a integração entre a computação em nuvem, redes inteligentes e a infraestrutura e serviços da IoT. Desta forma, devido a suas funcionalidades, a publicidade do código fonte de seus componentes e sua proposta de baixo custo de implantação, a plataforma Fiware pode emergir rapidamente no mercado.

Projeto SOFIA

O projeto SOFIA(*Smart Objects For Intelligent Applications*) fornece uma plataforma multi-linguagem, multi-protocolo e de código fonte aberto para a implementação de aplicações em ambientes inteligentes da IoT [Sofia, 2016a]. Um projeto desenvolvido pela organização Artemis[6], o SOFIA dispõem de uma plataforma de *middleware* que permite a interoperabilidade

[6] https://artemis-ia.eu/about_artemis.html

dos vários sistemas e dispositivos existentes na IoT. Além disso, o projeto também oferece uma plataforma semântica que permite a troca de informações entre os dispositivos do ambiente físico e as aplicações do mundo virtual. Assim, a plataforma SOFIA provê mecanismos para a publicação, monitoramento e organização de sensores e atuadores existes no ambiente físico.

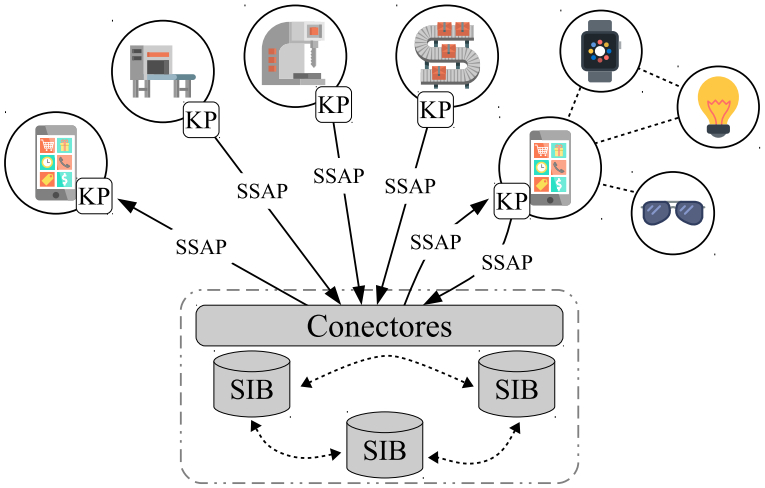


Figura 3.9: Conceito funcional SOFIA, baseado em [Sofia, 2016a]

A estrutura funcional da plataforma SOFIA se divide em processadores do conhecimento (KP), corretores de informações semânticas (SIB) e protocolo de acesso ao espaço inteligente (SSAP), como apresentado na Figura 3.9. O KP (do inglês, *Knowledge Processor*) processa os pacotes de qualquer dispositivos, aplicação ou sistema que produz ou consome uma determinada informação na plataforma. A SIB (do inglês, *Semantic Information Brokers*) mapeia, armazena e íntegra semanticamente as informações trocadas entre os KP's. O SSAP (do inglês, *Smart Space Access Protocol*) fornece a *interface* para execução de operações como, por exemplo, inserir, sair, juntar e remover na rede semântica. Essas funções definem os principais conceitos utilizados em sua arquitetura.

A plataforma do projeto utiliza módulos para compor sua arquitetura. Essa arquitetura modular está composta por módulos como, *SIB Runtime*, *SIB Tools*, *Integral DB*, *SIB Web Console*, *SDK* e *SIB API Manager*. A *SIB Runtime* oferece a *interface* de comunicação para diferentes clientes e protocolos. A *SIB Tools* processas as solicitações que chegam no módulo anterior. O módulo *Integral DB* está divido em CDB (*Configuration Database*) responsável por armazenar os dados de configuração da plataforma, RTDB (do inglês, *Real Time Database*) armazena as instâncias de ontologias recebidos dos KP durante um período de tempo e HDB (*Historic Database*) que armazenam todos os dados não utilizados em tempo real. O módulo *SIB Web Console* provê uma *interface web* simples para gerenciar e configuras a plataforma. Os módulos *SDK* fornece API's em várias linguagem e ferramentas para o desenvolvimento ágil de aplicações. O *SIB API Manager* possibilita o gerenciamento das API's desenvolvidas para os clientes. Assim, sua arquitetura modular proporciona uma compatibilidade com as principais plataformas em nuvem, existentes na atualidade.

Deste modo, o projeto SOFIA torna possível o acesso a dados do mundo real em ambientes virtuais. Esses dados coletados no ambiente físico provê as informações necessárias para o desenvolvimento de serviços para domínios inteligentes como *Smart Home* (Casa Inteligente) e *Smart City* (Cidade Inteligente). Assim, esta plataforma permite utilizar o grande volume de dados gerados por coisas da IoT em conjunto com a infraestrutura elástica da computação em nuvem para o desenvolvimento ágil de serviços e aplicações.

Projeto Nimbits

O projeto de código aberto da Nimbits [Nimbits, 2016] fornece uma plataforma de conectividade M2M. A plataforma disponibiliza bibliotecas em Java como, *nimbits.io*, para o desenvolvimento de soluções para IoT e nuvem. Essas soluções permitem a criação de uma plataforma como serviço que provê a coleta, processamento, gravação e compartilhamento dos dados de sensores da IoT. Desta forma, ela fornece meios para o desenvolvimento de soluções de *hardware* e *software* que integram os dispositivos da IoT com a nuvem. Além disso, essas soluções permitem o armazenamento e a recuperação do grande volume de dados gerados pelos dispositivos físicos contidos na IoT.

Sua plataforma possui uma estrutura com árvore de entidades. As entidades da árvore dispõem de um nome, ID único e um pai. Essas entidades baseiam-se em registros de dados e tecnologias baseadas em regras. O registro de dados são armazenados em pontos de dados, que consistem de um conjunto de valores armazenados em formatos como JSON ou XML. Além disso, os pontos de dados podem ser configurados para gerar regras como cálculo, estatística, alerta ou transmitir dados importantes para a nuvem. Desta forma, a plataforma tem um baixo grau de complexidade no desenvolvimento de aplicações para a IoT.

A plataforma Nimbits foi projetada para funcionar em dispositivos com recursos limitados. Seu projeto utiliza protocolos como XMPP [Karagiannis et al., 2015] na troca de mensagens entre seus serviços. Além disso, a plataforma permite gerenciar e desenvolver seus serviços via *interface web*. Desta forma, suas aplicações podem ser executados em pequenos dispositivos como, *Raspberry Pi* [Raspberry, 2017b] e em nuvens como, *Google App Engine* [Google, 2017] e *Amazon EC2*[Amazon, 2017]. Assim, a plataforma permite filtrar ruídos dos sensores e transmitir seus dados até grandes servidores.

Projeto LinkSmart

Uma plataforma desenvolvida no âmbito do projeto Hydra[7] e co-financiado pela Comissão Europeia, o LinkSmart® [Kostelnik et al., 2011] desenvolve uma estrutura de serviços com arquitetura distribuída para dispositivos heterogêneos. Sua plataforma de *midleware* dispõem de ferramentas que possibilitam a detecção, incorporação e utilização de qualquer dispositivo contido na IoT. Desta forma, o projeto LinkSmart soluciona os problemas de incompatibilidade de protocolos entre os dispositivos. Além disso, suas ferramentas de desenvolvimento permitem a construção de aplicações inteligentes que controlam através de serviços *web* qualquer dispositivo físico distribuído nos ambientes inteligentes da IoT.

A plataforma LinkSmart® utiliza em suas soluções uma combinação de tecnologia de serviços de *web* semântica com princípios baseados em aplicações SOA. O SOA permite alcançar a interoperabilidade ao nível sintático, ou seja, especifica padrões para a troca de dados e serviços entre os dispositivos. Os serviços de *web* semântica em conjunto ao uso de ontologias proporciona a obtenção de interoperabilidade semântica, isto é, mapeia as informações dos objetos físico e gera o conteúdo com significado em comum entre os dispositivos. Desta forma, o projeto LinkSmart facilita o desenvolvimento de aplicações e provê acessibilidade uniforme em seus serviços *web*.

As aplicações da plataforma LinkSmart são construídas fundamentalmente como serviços *web*. A plataforma representa todos os dispositivos como um serviço *web*. Esta abstração permite incorporar objetos a sua plataforma sem se preocupar com tecnologias de rede como, RFID [Jia et al., 2012], Zigbee [Gungor e Hancke, 2009], e Bluetooth [Chang, 2014]

[7] http://hydramiddleware.eu/news.php

utilizadas por estes dispositivos. A plataforma também dispõem de gestores centrais que fornecem funcionalidades úteis para serviços *web* como, funcionalidades para o desenvolvimento de aplicações inteligentes. Esses gestores estão fracamente vinculados ao *middleware*. Esta ligação fraca permite aos dispositivos com restrições de *hardware* executar fragmentos do *middleware* sem a necessidade de servidores centrais. Assim, a arquitetura do projeto LinkSmart permite que dispositivos com limitações cooperem na execução de aplicações complexas.

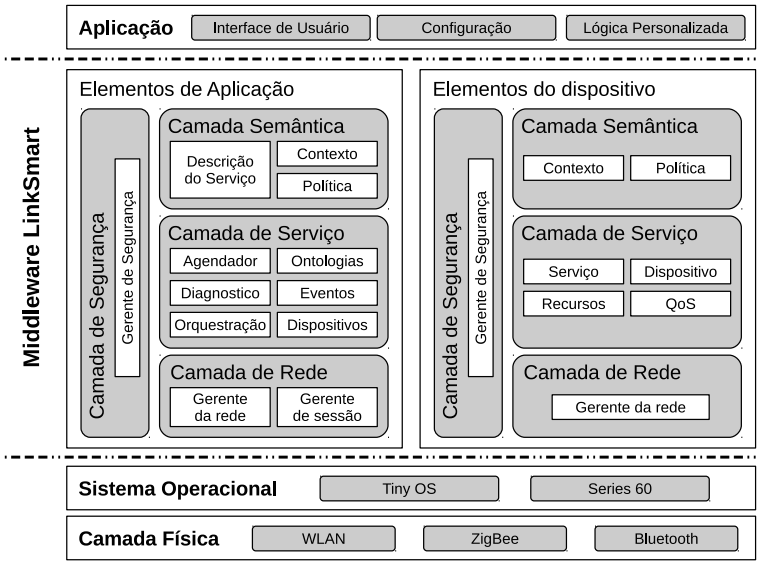


Figura 3.10: Arquitetura LinkSmart® [Kostelnik et al., 2011]

O projeto LinkSmart possui uma arquitetura divida em camada física, sistema operacional, *middleware* e aplicação, apresentado na Figura 3.10. A camada física fornece a *interface* de conexão para os dispositivos. A camada de sistema operacional gerencia os objetos da camada inferior e fornece métodos para os recursos de conexão de rede. Na camada de aplicação estão as aplicações desenvolvidas pelos usuários, tais como, gerenciamento de fluxo. O *middleware* resume-se em dois elementos, sendo eles, aplicação e dispositivo. O elemento aplicação se encarrega do desempenho dos recursos utilizados e o elemento dispositivo fica responsável pelo acesso remoto. Assim, ambos os elementos possuem funções como, detecção de contexto, solicitação de serviço de transporte, gerenciamento da rede, sincronização, controle de acesso, e entre outros. Desta forma, a plataforma LinkSmart permite a criação de uma rede para sistemas embarcados com descoberta de dispositivo via semântica.

Plataforma ThingSpeak

O projeto ThingSpeak desenvolve uma plataforma de aplicações descentralizada com código aberto (*Open-source*) para a IoT [ThingSpeak, 2016a]. Esta plataforma permite facilitar o gerenciamento de serviços disponíveis a partir dos dispositivos heterogêneos da IoT. Desta forma, sua plataforma permite coletar, armazenar, analisar, visualizar e agir sobre os dados gerados por sensores e atuados como, por exemplo, *Arduino*® [Arduino, 2017] e *Raspberry Pi* [Raspberry, 2017b]. Assim, o projeto ThingSpeak proporciona coleta em tempo real, processamento e visualização dos dados de dispositivos integrados a sua plataforma.

A plataforma ThingSpeak disponibiliza um servidor para o armazenamento e recuperação de dados da Internet das coisas. Este servidor permite a abertura de canais de fluxo de dados com suporte a formatos de dados em JSON e XML para os usuários. Além disso, ela também dispõe de ferramentas de visualização que permitem a criação de *widgets* em linguagens de

desenvolvimento *web* como, JavaScript, HTML e CSS. Assim, essas *widgets* possibilitam uma visualização personalizada dos dados coletados por uma aplicação.

O desenvolvimento de aplicações na plataforma ThingSpeak utiliza a *API ThingSpeak*. Esta API possui ferramentas para o processamento de solicitações HTTP, armazenamento de dados numéricos e alfanuméricos, processamento de dados numéricos, rasteamento de localização, e atualização de *status*. Assim, sua estrutura permite o desenvolvimento de uma infinidade de serviços para a IoT como, aplicações de rastreamento de localização e rede social de coisas com atualizações de *status*, podendo, por exemplo, controlar o termostato de sua com base em sua localização atual.

3.4.4 Análise das Plataformas Baseadas em CC

Todas as plataformas descritas oferecem estruturas para o desenvolvimento de aplicações em ambientes da IoT. Elas dispõem de ferramentas que facilitam o desenvolvimento de serviços e aplicações, pois, a maioria dispõe de interface web. Além disso, as plataformas atendem a necessidades de heterogeneidade e interoperabilidade existente na rede. No entanto, suas estruturas não possuem soluções que realizem filtragem e encaminhamento dos fluxos de dados na *Fog*. As análises feitas em [Mineraud et al., 2016] mostram que as plataformas baseadas em nuvem, futuramente, incluirão soluções de apoio ao processamento de fluxos de dados na *Fog* em seu conjunto de ferramentas.

A Tabela 3.1 lista as plataformas pesquisadas e resume as características consideradas fundamentais para atender as expectativas dos usuários e desenvolvedores de aplicação. A coluna (a) apresenta os tipos de dispositivos suportados pela plataforma. Esse suporte permite integrar qualquer dispositivo da IoT, independente da tecnologia utilizada. Entre as plataformas, somente a LinkSmart possui limitações no suporte à heterogeneidade dos dispositivos da IoT. Essa limitação impede a plataforma de trabalhar com tecnologias emergentes, ocasionando problemas na adoção de novos protocolos e suporte ao número crescente de dispositivos heterogêneos.

Plataforma	**a) Sup. Disp. heterogeneos**	**b) REST**	**c) Descob. de serviço**	**d) Processam. local**	**e) Interoperab. dos dados**	**f) Fog**
Fiware	Sim	Sim	Sim	Sim	NGSI	Não
LinkSmart	Dispositivos embarcados	Não	Sim	Sim	Semântica	Não
Nimbits	Sim	Sim	Não	Não	Pontos de dados	Não
OpenIoT	Sim	Não	Sim	Não	Semântica	Não
Sofia	Sim	Sim	n/e	Não	Semântica	Não
ThingSpeak	Sim	Sim	Limitado	Não	Canal	Não

Tabela 3.1: Comparação entre as plataformas, adaptado de [Mineraud et al., 2016].

A coluna (b) apresenta quais as plataforma que possuem suporte ao protocolo de comunicação REST [Karagiannis et al., 2015]. As plataformas LinkSmart e OpenIoT não dispõem de suporte à tecnologia REST. Esse protocolo utiliza os métodos HTTP[8] *get*, *post*, *put* e *delete* para fornecer um sistema de mensagens, onde todas as ações podem ser executadas através de comandos HTTP de solicitação/resposta. O REST possui papel importante do IoT, pois, é

[8] Protocolo de camada de aplicação da Web, define o formato e a sequência das mensagens que são passadas entre o navegador e o servidor [Kurose e Ross, 2012].

suportado por muitas plataformas de nuvem M2M. Além disso, ele pode ser implementado facilmente em aplicativos de *smartphone* e *tablet*.

A coluna (c) resume a característica referente ao suporte à descoberta de serviço. Essa característica é um ponto importante na IoT, pois, permite detectar automática de objetos e serviços disponíveis em ambientes da IoT. Assim, a plataforma consegue encontrar os objetos e os serviços disponíveis na IoT, mesmo com a alta mobilidade dos objetos. Na comparação entres as plataformas pesquisadas, as plataformas ThingSpeak, Nimbits e Sofia obtiveram resultado diferente das demais. A ThingSpeak apresenta limitações e a Nimbits não despõe de suporte à descoberta de serviço. Já, a Sofia não específica em sua documentação se possui ou não suporte à descoberta de serviço.

O processamento local, apresentado na coluna (d), especifica se a plataforma dispõe de mecanismo que efetuem o processamento de dados próximo do usuário final. Essa característica permite reduzir o envio de dados brutos aos centros de dados dos provedores de serviços em nuvem. Entre as plataformas pesquisadas, somente as plataformas Fiware e LinkSmart dispõe de mecanismo para processamento local de dados. A interoperabilidade dos dados, resumido na coluna (e), refere-se ao processo de dar sentido único às informações da IoT. Assim, esse sentido único permite aos consumidores de informações da IoT entender e processar os dados. Cada plataforma possui um método de interoperabilidade de dados, as plataformas LinkSmart, OpenIoT e Sofia utiliza semântica para organizam e disponibilizar seus dados. A Fiware usa entidade de contexto contida na tecnologia NGSI [Moltchanov e Rocha, 2014] e a Nimbits armazena seus dados em pontos de dados, que consistem de um conjunto de valores em formatos JSON ou XML. Já, a ThingSpeak utiliza canais de fluxos de dados com suporte a formatos JSON e XML.

A coluna (f) resume quais plataformas possuem suporta à *Fog*. Essa característica permite utilizar os recursos computacionais disponíveis na borda de rede. O uso desses recursos possibilita o encaminhamento, filtragem, processamento e armazenamento de dados próximos do usuário final. Desta forma, os dados não precisam ser direcionados aos provedores de serviço em nuvem. Dentre as plataformas pesquisadas, até o presente momento, nenhuma das plataformas dispõe de suporte à *Fog*. Por fim, a partir das comparações feitas na Tabela 3.1 conclui-se que a plataforma Fiware atende a mais requisitos que as demais plataformas.

3.5 Considerações Finais

O capítulo apresenta algumas soluções que buscam solucionar as necessidades no gerenciamento de dados na IoT. Essas soluções estavam classificadas em três abordagens, sendo elas, CC, Fog e SDN. A abordagem em CC mostrou plataformas de código aberto que buscam desenvolver soluções que integram os recursos elásticos da computação em nuvem com a IoT. A abordagem em Fog apresenta algumas soluções que propõem processar, armazenar e gerenciar os dados mais próximo da fonte, diminuindo o tráfego desnecessário e a latência na rede. As abordagens em SDN mostrou os projetos que utilizam a filosofia de flexível da SDN para controlar os fluxos de dados da IoT. Portanto, este capítulo mostrou algumas soluções que buscam gerenciar os fluxo de dados da IoT. Assim, no próximo capítulo será apresentado a proposta deste trabalho para efetuar o gerenciamento de fluxo de dados na IoT.

Capítulo 4

Abordagem para Gerenciamento de Fluxo de Dados da IoT na Fog

O conceito da IoT têm como objetivo fornecer dados para o desenvolvimento de serviçõe e aplicações. No entanto, a grande quantidade de dispositivos inseridos na IoT aumentam consideravelmente a velocidade e o volume de dados gerados. Além disso, como apresentado no Capítulo 2, os dispositivos contidos na IoT possuem restrições de recursos e heterogeneidade de *hardware*. Estas características agregam uma maior complexidade no desenvolvimento de serviços na IoT. Por este motivo, como apresentado no Capítulo 3, a integração entre Nuvem e IoT através de plataformas baseadas em Nuvem traz benefícios para a IoT, por exemplo, processamento ilimitado, armazenamento em larga escala e soluções para conectar, controlar e gerenciar qualquer coisa em qualquer lugar a qualquer momento [Botta et al., 2016]. No entanto, o tráfego dos dados gerados na IoT para os centros de dados da Nuvem pode agregar uma maior latência nos serviços.

Diante desse cenário, apresenta-se uma abordagem para gerenciamento de fluxos de dados da internet das coisas na Fog em conjunto com as plataformas para a IoT baseadas na Nuvem. Essa solução proposta utiliza os recursos disponíveis na Fog para filtrar, agregar e encaminhar os fluxos de dados da IoT na borda da rede. Assim, os dados são processados próximo de sua origem e, consequentemente, minimiza a quantidade de dados encaminhados para os centros de dados da Nuvem.

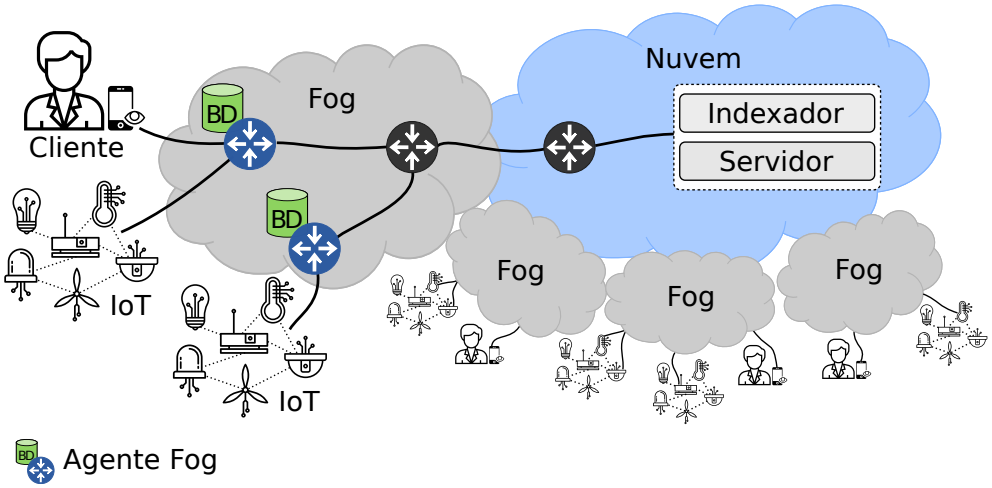


Figura 4.1: Visão geral da abordagem

A Figura 4.1 ilustra a visão geral do ambiente de atuação da abordagem proposta. O ambiente é dividido em duas partes, a nuvem e a *Fog*. A nuvem dispõe de um conjunto elástico de recursos computacionais. Além disso, na nuvem existem dois dispositivos importantes para o funcionamento da abordagem proposta, os *servidores* e o *indexador*. Os *servidores* são responsáveis por atenderem as requisições dos clientes, caso os recursos computacionais existentes na *Fog* não sejam suficientes. O *indexador* são servidores responsáveis por fornecer uma lista com os dispositivos contidos nos ambientes da IoT. Ele mantém uma lista que metodicamente indica a localização e conteúdo disponível em objetos contidos na IoT. Em contrapartida, o ambiente da *Fog* é composta por dispositivos existentes na borda da rede. Neste trabalho, os dispositivos são denominados genericamente de *agente Fog* (AF). Os AF fornecem meios que tornam possível filtrar e encaminhar os dados gerados pelos dispositivos da IoT.

O restante do capítulo apresenta as especificações da abordagem para gerenciamento de fluxo de dados da IoT na *Fog*. A Seção 4.1 descreve o modelo da rede, mostrando o funcionamento da solicitação e encaminhamentos dos dados. A Seção 4.2 descreve a abordagem para gerenciamento de fluxo de dados da IoT na *Fog* e suas etapas de funcionamento. A Seção 4.3 apresenta as considerações finais do conteúdo abordado no capítulo.

4.1 Modelo da Rede

O modelo de rede considerado possui ambientes que comportam os objetos (coisas) da IoT. Esses objetos interagem entre si de modo a transmitir os dados coletados. A Figura 4.2 ilustra o modelo de rede considerado neste trabalho, apresentando cada elemento que compõe a rede. Na Figura 4.2(a) estão contidos os ambientes da IoT (IoT_y e IoT_x) e o cliente (C_n). Os ambientes da IoT possuem em sua extensão um conjunto de objetos $n = \{n_1, n_2, n_3, ..., n_i\}$ que coletam e encaminham as informações geradas no mundo real. No que lhe concerne, o nó cliente (C_n) consome os fluxos de dados gerados nos ambientes da IoT. Na Figura 4.2(b) contém o ambiente da *Fog*. Este ambiente possui um conjunto de dispositivos $g = \{g_1, g_2, ..., g_n\}$, denominados agente *Fog* (AF). Eles são responsáveis por intermediar a comunicação dos objetos da IoT com a Internet. Por fim, a Figura 4.2(c) ilustra os serviços providos no ambiente da nuvem.

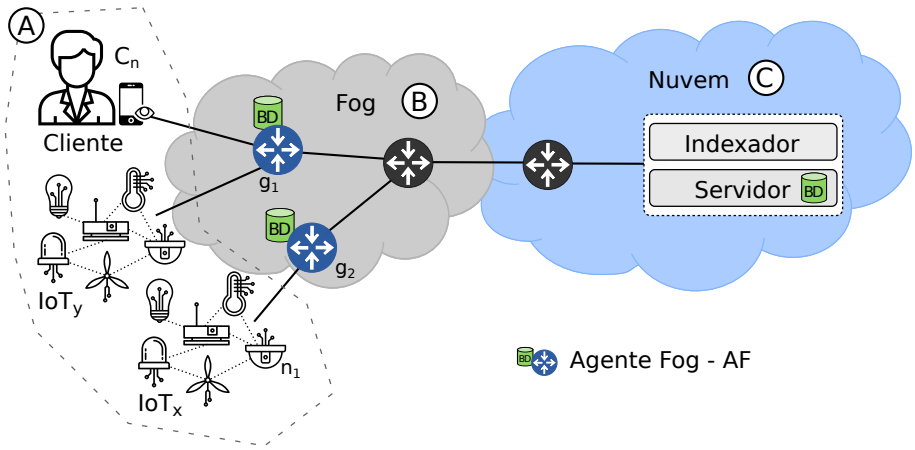
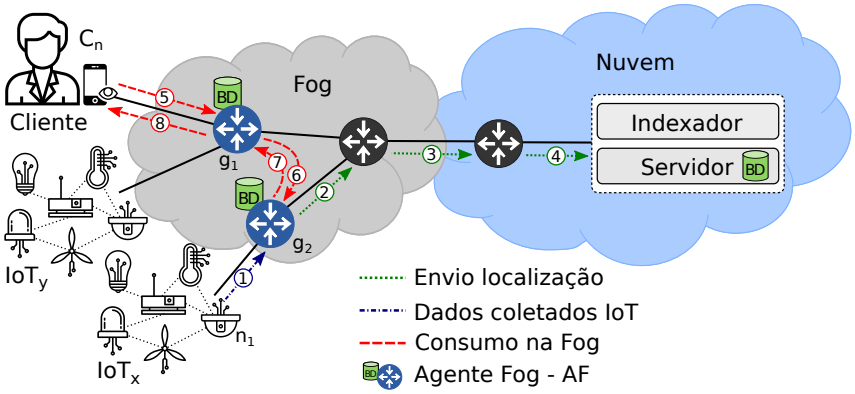


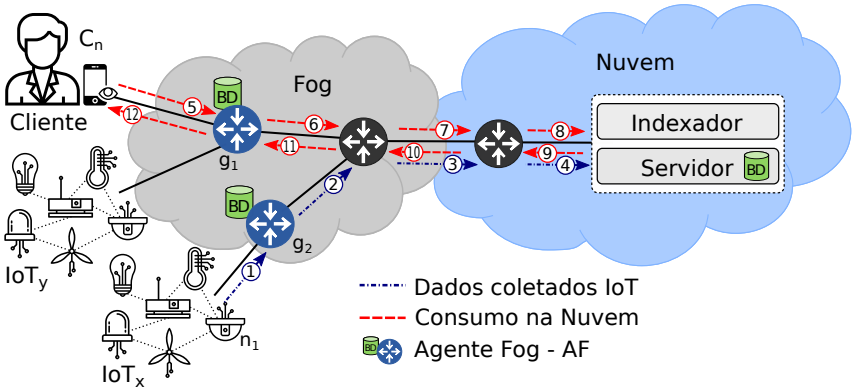
Figura 4.2: Modelo da rede

A Figura 4.3 ilustra o tráfego do fluxo de dados contido na rede. O encaminhamento do fluxo de dados na rede ocorre de duas formas, na *Fog* ou para a *Nuvem*. Para exemplificar será utilizado uma solicitação fictícia de fluxo de dados gerados pelo objeto n_1 contido no ambiente IoT_x. A Figura 4.3(a) apresenta o tráfego de dados na *Fog*. O objeto n_1 encaminha os dados

para o agente *Fog* g_2 (seta 1). O AF g_2 encaminha a localização dos objetos que constam em sua base de dados para o *indexador* contido na nuvem (seta 2, 3 e 4). Para consumir os dados gerados por n_1, o cliente C_n efetua a solicitação, através da aplicação, para o AF em que possui conexão, no caso o AF g_1 (seta 5). Neste exemplo partimos do princípio que o AF g_1 esteja ciente da localização de n_1, sem a necessidade de realizar uma consulta ao *indexador*. O AF g_1 encaminha a solicitação para o AF g_2 (seta 6). Por fim, o AF g_2 encaminha os dados solicitados pelo cliente (seta 7 e 8).



(a) Consumo na Fog



(b) Consumo na Nuvem

Figura 4.3: Fluxo de dados

A Figura 4.3(b) ilustra o tráfego de dados na Nuvem. O objeto n_1 encaminha as informações para o agente *Fog* g_2 (seta 1). O AF g_2 processa as informações e encaminha para os servidores contidos na nuvem (seta 2, 3 e 4). O cliente C_n solicita para o AF g_1, através da aplicação, os dados gerados por n_1 (seta 5). O AF g_1 encaminha a solicitação para a nuvem (seta 6, 7 e 8). Por fim, a nuvem responde a requisição e encaminha os dados solicitados (seta 9, 10, 11 e 12).

4.2 Descrição da Abordagem

A abordagem proposta para o gerenciamento de fluxos de dados da Internet das Coisas na Fog está dividida em três componentes, *agente Fog*, *appCliente* e *indexador*. Apresentado na Figura 4.4(b), o *agente Fog* é responsável por tornar acessível, manter uma base de dados e encaminhar os dados gerados na IoT. Esse agente se divide em quatro componentes: *gerenciador*, *base*, *encIndex* e o *encUs*. O *gerenciador* minimiza as complexidades contidas na heterogeneidade

dos objetos existentes na IoT. Além disso, ele disponibiliza meios para tornar os objetos contidos na rede disponíveis, acessíveis e utilizáveis. A *base* mantêm uma base de dados dos objetos da IoT e seus respectivos conteúdos de monitoramento do ambiente físico. O *encUs* é responsável por receber as requisições e encaminhar para o solicitante os dados especificados nas requisições. O *encIndex* encaminha para o indexador contido na nuvem a localização e os sensores existentes em um determinado objeto.

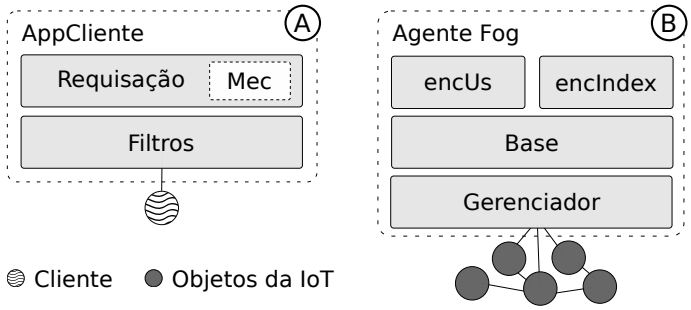


Figura 4.4: Componentes da abordagem

No *appCliente*, apresentado na Figura 4.4(a), o cliente especifica, através de filtro, o sensor e efetua a requisição das informações que deseja consumir. Além disso, esse componente possui um mecanismo de controle de atraso. Esse mecanismo monitora o tempo de resposta do *agente Fog* e caso o atraso seja alto, o *appCliente* direciona a requisição para os servidores da nuvem. O *indexador* mantêm na nuvem uma lista que indica metodicamente a localização dos objetos contidos nos *agentes Fog*.

4.2.1 Funcionamento

A partir do cenário existente na Figura 4.2, suponha que o cliente queira realizar a aquisição de dados gerados por um objeto contido no ambiente da IoT. Além disso, considere que o cliente tenha estabelecido comunicação com no mínimo um *agente Fog* (AF). No cenário assumimos que o processo de solicitação dos dados pode ser encerrado de três maneiras distintas. No *Caso 1*, o cliente obtém dados direto no AF em que possui comunicação. No *Caso 2*, o AF encaminha mensagem de erro para o cliente, pois, o AF e nem o *indexador* existente em nuvem conhecem a localização do objeto. Por fim, no *Caso 3*, o AF não possui o objeto em sua base, mas encontra a localização do mesmo no *indexador*. A Figura 4.5 ilustra, através de fluxograma, o processo de solicitação dos dados:

Caso 1: o cliente obtém os dados direto no AF em que possui comunicação:

1. Inicialmente, o cliente utiliza o *appCliente* para solicitar o consumo de dados de um determinado objeto da IoT. Por meio da aplicação, o cliente solicita para o AF os dados coletados pelo sensor.

2. O AF recebe a solicitação, verifica se o sensor existe em sua *base*, processa os dados e posteriormente os encaminha para o cliente.

Caso 2: o AF encaminha mensagem de erro para o cliente, pois, o AF e nem o *indexador* contido na nuvem conhecem a localização do objeto:

1. O cliente solicita por meio da aplicação o consumo de dados de um determinado sensor da IoT. A aplicação envia a solicitação para o AF em que possui conexão.

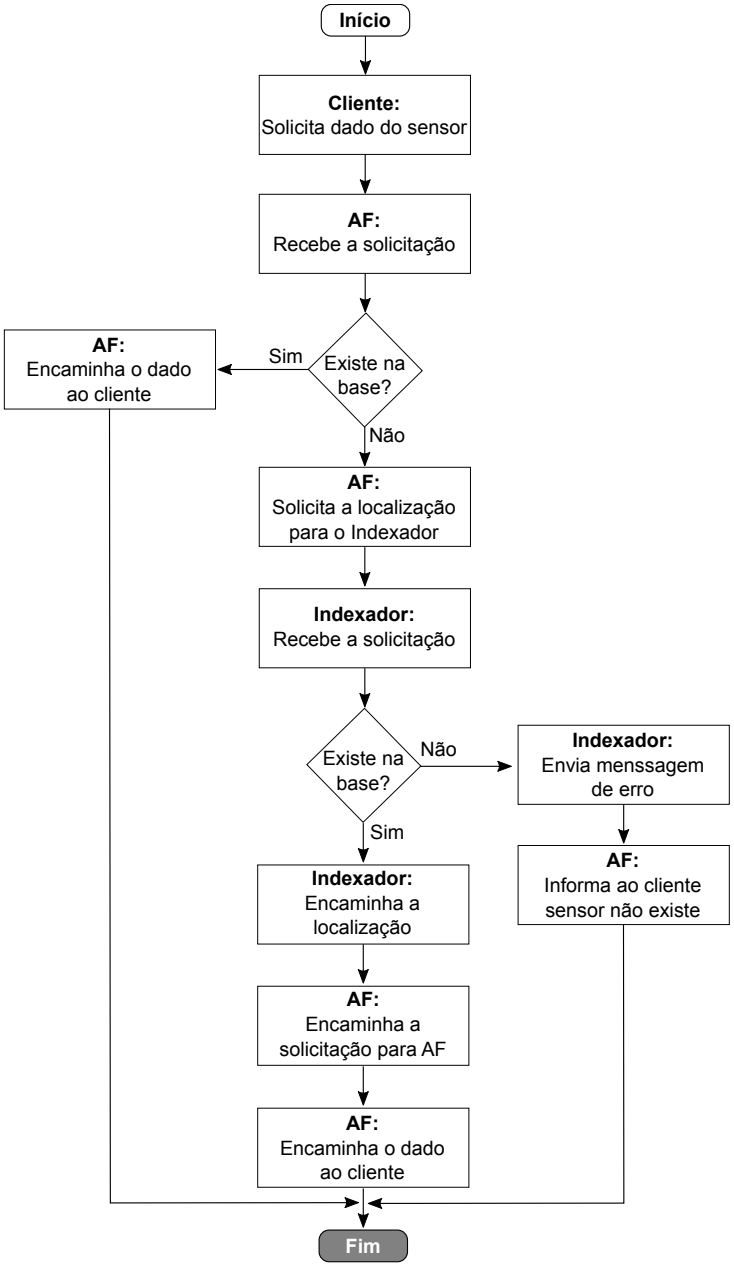


Figura 4.5: Fluxograma do funcionamento

2. O AF recebe a solicitação e verifica se o sensor existe em sua *base*. O AF não possui em sua *base* o sensor, mas, consequentemente encaminha uma requisição para o *indexador* contido na nuvem solicitando a localização do sensor em questão.

3. O *indexador* verifica a existência do sensor em sua base, mas, não consta na base. Então o *indexador* envia uma mensagem de erro para AF.

4. O AF encaminha a mensagem de erro para o cliente.

Caso 3: o AF não possui o objeto em sua base, mas o encontra a localização do mesmo no *indexador*:

1. Inicialmente, o cliente utiliza o *appCliente* para solicitar o consumo de dados de um determinado objeto da IoT. A aplicação envia a solicitação para o AF em que está diretamente conectado.

2. O AF recebe a solicitação e verifica se o sensor existe em sua *base*. O sensor não consta na base. Então o agente encaminha uma requisição para o *indexador* solicitando a localização do sensor em questão.

3. O *indexador* verifica a existência do sensor em sua base e posteriormente responde a requisição com a localização do AF que tenha em sua *base* o sensor em questão.

4. O AF (agente *Fog* em que o cliente possui comunicação) encaminha a solicitação para o AF (agente *Fog* informado pelo *indexador*).

5. O AF (agente *Fog* informado pelo *indexador*) encaminha os dados solicitados para o cliente.

O processo de solicitação para o *indexador* ocorre somente quando o *agente Fog* não possui a localização do sensor. Após obter esta localização, todas a demais solicitação de dados são efetuadas nos domínios da *Fog*. Desta forma, a latência na primeira solicitação será mais elevada, pois, existe a necessidade de buscar na nuvem a localização do sensor que o cliente deseja consumir os dados. No entanto, a latência nas requisições subsequentes será inferior à obtida na primeira, pois, essas requisições ocorreram somente na Fog.

Uma das principais vantagens do encaminhamento de dados na borda de rede é a baixa latência. No entanto, a *Fog* é composto por dispositivos que não dispõem de recursos computacionais ilimitados. Essas limitações podem ocasionar atraso no encaminhamento dos dados quando os dispositivos passam por algum estresse, por exemplo, um nó AF pode receber uma grande quantidade de solicitações. Essas solicitações exigiriam uma quantidade maior de processamento do dispositivo (CPU e memória) para serem atendidas. Assim, o estresse sobre o *hardware* aumentaria o tempo de resposta do nó AF para o cliente.

Para amenizar o estresse sobre o *hardware* e, manter uma baixa latência, a aplicação direciona as solicitações para a nuvem. O Algoritmo 1 apresenta a sequência lógica do mecanismo de prevenção do atraso. Ao solicitar o dado do sensor, a aplicação mantém um *timeout* com o tempo máximo de espera da resposta vinda do nó AF. Se tempo de resposta do nó AF for inferior ao tempo estipulado no *timeout*, o cliente mantém as solicitações na *Fog*. Caso contrário, se o tempo de resposta for maior que o *timeout*, o cliente direciona as solicitações para os servidores da nuvem. Assim, a aplicação minimiza a quantidade de novas solicitações direcionadas ao AF sobrecarregado.

Algoritmo 1 Mecanismo de prevenção do atraso:

```
1: if SolicitaDadoFog(Sensor) > timeout then
2:     dadosSensor ← SolicitaDadoNuvem(Sensor)
3:     return dadosSensor
4: else
5:     return SolicitaDadoFog(Sensor)
6: end if
```

4.2.2 Implementação

Esta seção apresenta as especificações de implementação da abordagem proposta neste trabalho. Alguns elementos (por exemplo, o *agente Fog*) utilizam habilitadores genéricos disponíveis no projeto Fiware [Fiware, 2017b]. Esses habilitadores genéricos são componentes funcionais que permitem aos desenvolvedores implementar funcionalidades no ecossistema

Fiware. Além disso, a abordagem proposta também possui elementos (por exemplo, o *appCliente*) que foram desenvolvidos em *python* 3.5.

Na Nuvem, ilustrada na Figura 4.2(c), estão localizados os serviços do *indexador* e *servidor*. O *indexador* foi modelado em MongoDB versão 3.0.15, uma aplicação de código aberto com alto desempenho que fornece uma base de dados orientado a documentação [MongoDB, 2017]. O *servidor* é responsável por armazenar os dados e atender as requisições dos clientes quando o *agente Fog* estiver sobrecarregado. Nele foi implementado o habilitador genérico *Orion ContextBroker* [Fiware, 2017a] do projeto Fiware. O *Orion*, através das *interfaces* NGSI9 e NGSI10 [Moltchanov e Rocha, 2014], utiliza um sistema de publicação/assinatura (*Publish/-Subscribe*) para registrar os objetos, atualizar informações e enviar notificações quando ocorrer alteração na informação ou notificar em intervalo de tempo pré-estabelecido.

O *agente Fog*, apresentado na Figura 4.4(b), é responsável por tornar acessível, manter uma base de dados e encaminhar as informações geradas na IoT. O componente *gerenciador* utiliza o habilitador genérico Fiware IoT [Fiware, 2016] para torna os objetos contidos na rede, disponíveis, pesquisáveis, acessíveis e utilizáveis no desenvolvimento de aplicações. O Fiware IoT, descrito na subseção 3.4.3 do capítulo 3, possui suporte a vários protocolos existentes na IoT. Sua arquitetura modular, denominada de *Agente IoT*, reduz as complexidades na integração dos objetos. No entanto, para associar o objeto à rede é necessário informar o protocolo utilizado na comunicação para selecionar o *agente IoT* correto. Os componentes *base* e *encUs* utiliza o *Orien ContextBroker* para traduzir os recursos da IoT para entidades de contexto NGSI [Fiware, 2017a]. Assim, os desenvolvedores acessam os dados da IoT como atributos de entidades representativas e podem enviar comandos, através de atualização atributos, relacionados a dispositivos. O componente *encIndex* foi desenvolvido em python 3.5. Esse componente utiliza o protocolo de comunicação REST[1] para encaminhar ao *indexador* um arquivo JSON[2] com a localização e os atributos de cada objeto contido no *agente Fog*.

A *appCliente*, ilustrada na Figura 4.4(b), é a aplicação cliente responsável por consumir os dados do *agente Fog*. Essa aplicação foi desenvolvido em *python* 3.5, com o uso da biblioteca *pycurl* para efetuar requisição via protocolo de comunicação REST. O cliente efetua o *filtro*, indicando o ID do objeto que deseja consumir os dados. Em seguida, a aplicação realiza a *requisição* dos dados do objeto escolhido. Assim, o cliente solicita os dados dos objetos disponíveis no elemento *Orien ContextBroker* contido no *agente Fog*.

4.3 Considerações Finais

Este capítulo apresentou uma descrição da abordagem proposta. Inicialmente foi apresentado uma visão geral da solução, que consiste em uma abordagem de gerenciamento de fluxo de dados da IoT na borda da rede. Esta solução possibilita aos clientes consumir com uma baixa latência os dados gerados nos ambientes da IoT. Essa baixa latência foi obtida através da utilização dos recursos disponíveis próximo do usuário final, ou seja, na *Fog*. O capítulo também descreveu o modelo de rede considerado pelo trabalho, mostrando o funcionamento das solicitações e encaminhamentos dos dados tanto na Nuvem quanto na Fog. Além disso, foi apresentada a descrição da abordagem proposta e suas etapas de funcionamento, incluindo cada etapa de execução e os possíveis casos que podem ocorrer durante o uso da abordagem. Por fim, na última subseção efetuou a descrição de como a abordagem foi implementada.

[1] Um estilo arquitetural que consiste de um conjunto coordenado de restrições arquiteturais aplicadas a componentes, conectores e elementos de dados dentro de um sistema de hipermídia distribuído [Karagiannis et al., 2015].

[2] Formato de texto leve para troca de dados, constituído de pares atributo-valor.

Capítulo 5

Validação dos Resultados

Neste capítulo são efetuadas as avaliações, por meio de simulação, a abordagem proposta contida no capítulo antecedente. A Seção 5.1 apresenta o modelo de simulação e a metodologia utilizada na experimentação. A Seção 5.2 descreve as métricas utilizadas para validar os experimentos ocorrido durante a simulação. A Seção 5.3 apresenta os resultados obtidos nas experimentações efetuadas. A Seção 5.4 mostra um resumo do conteúdo abordado no capítulo.

5.1 Modelo Simulação

Um ambiente de simulação composto por uma combinação de dispositivos reais e virtuais foi projetado e desenvolvido para validar a eficiência da abordagem proposta neste trabalho. Os dispositivos reais foram utilizados com o intuito de fornecer maior grau de realidade na perspectiva dos dispositivos. Enquanto isso, a adoção concomitante de dispositivos virtualizados possibilitou a realização de experimentos com número maior de dispositivos e, consequentemente, aumentar o grau de realidade na perspectiva da rede.

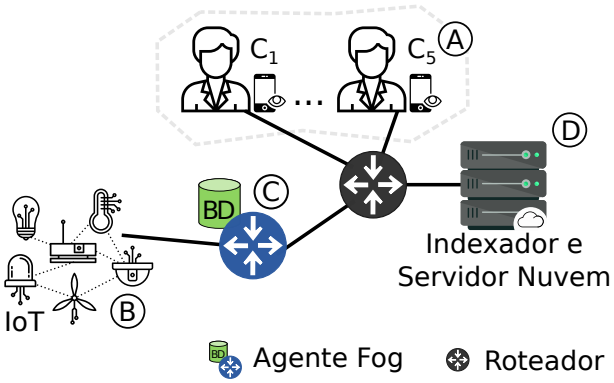


Figura 5.1: Cenário da validação

A Figura 5.1 ilustra os elementos contidos no cenário proposto para a validação. Esses elementos caracterizam o nó cliente, os dispositivos e serviços contidos nos ambientes da IoT, Fog e Nuvem. O nó cliente, apresentado na Figura 5.1(a), executa a aplicação *appCliente*. Ela foi implementada em computadores que dispõem de CPU Intel Core i5-2400 @ 3.4GHz e 6GB RAM. No total foram utilizados cinco computadores para solicitar dados aos *agentes Fog*. A Figura 5.1(b) apresenta os objetos contidos no ambiente da IoT. Os objetos foram simulados através da aplicação *Fiware Device Simulator* [Fiware, 2017c]. Essa aplicação simula os dispositivos e suporta o formato JSON via HTTP e MQTT, possibilitando executar simulações

em tempo real e em modos de avanço rápido. O simulador foi executado em um computador que dispõe de CPU Intel Core i5-4210U @ 3.4GHz e 8GB RAM. A comunicação dos objetos da IoT com o (*agente Fog*) ocorreu via tecnologia *Wi-Fi*.

O *agente Fog* (AF), ilustrado na Figura 5.1(c), representa os dispositivos existentes na *Fog*. O AF foi implementado na terceira geração de dispositivos desenvolvidos pela Fundação Raspberry [Raspberry, 2017b]. O *Raspberry Pi* 3 modelo B (Pi3) é um dispositivo popular e financeiramente acessível, podendo ser adquirido por R$180 reais no comércio eletrônico nacional. Este modelo possui as seguintes especificações de *hardware* [Raspberry, 2017c]:

- Processador: Broadcom BCM2837 Quad Core ARM Cortex-A53 1.2GHz 64 bits
- Memória RAM: 1GB LPDDR2 (900 MHz)
- Rede: Interface Ethernet 10/100Mbps e Wi-Fi 802.11 b/g/n 2.4GHz
- Bluetooth: Bluetooth 4.1 e Bluetooth Low Energy (BLE)
- Portas: 4x USB 2.0, 1x Full HDMI e 1x microUSB para alimentação
- Armazenamento: Cartão MicroSD 16GB SanDisk Ultra Classe 10
- Expansão: Conector GPIOs com 2x20 pinos, Interface Câmera (CSI) e Interface Display (DSI)

O Pi3 executa o Pi64, um sistema operacional experimental de 64 *bits* para o *Raspberry Pi* 3 [Amarni, 2017]. Esse sistema foi desenvolvido com base no Debian Stretch[1] e possui o Kernel Linux 4.11.2 [Raspberry, 2017a]. A escolha do sistema Pi64 ocorreu pela falta de uma versão oficial que seja executável em arquitetura 64 *bits* no Pi3. Até o momento, apenas a openSUSE possui uma versão oficial para *Raspberry Pi* 3 na arquitetura 64 *bits* [openSUSE, 2017]. Contudo, ao tentar executar o MongoDB no sistema foi encontrado um problema de incompatibilidade de biblioteca. Por este motivo, utilizado-se o sistema operacional Pi64 que possui, através do repositório Debian, uma versão estável do MongoDB.

Na Figura 5.1(d) contém os elementos providos via nuvem. Os serviços do *indexador* e *servidor* foram implementadas em entidades contidas na infraestrutura do *Google Cloud Platform*, uma suíte de computação em nuvem comercializada pelo Google [Google, 2017]. Nela, ambos os serviços são executados em uma instância de máquina de virtual com sistema operacional Linux CentOS versão 6[2]. Essa instância dispõe de 4 vCPUs, 14 GB de memória e 30 GB de disco de armazenamento. Por fim, a comunicação entre nó cliente, *agente Fog* e os serviços providos em nuvem ocorrem via *switch Edge-corE* ES4548C.

5.1.1 Metodologia

No ambiente desenvolvido para a validação, os clientes requisitam dados de sensores contidos no *agente Fog*. As solicitações foram realizadas com um número inicial de 1, 10 e 20 requisições por segundo. Posteriormente, as requisições foram aumentadas de 20 em 20 até alcançar as 100 requisições. Após a primeira centena, as requisições foram aumentadas de 100 em 100 até alcançar as 1000 requisições. Em seguida as requisições foram aumentadas de 200 em 200 até as 2000 requisições por segundo. Por fim, o número de requisições foram aumentadas de 1000 em 1000 até alcançar as 10000 requisições. O número de requisições direcionadas ao

[1] `https://www.debian.org/`

[2] A escolha do sistema operacional seguiu as especificações de utilização sugeridas pela plataforma Fiware.

agente Fog foi dividida entre os cinco computadores do ambiente de simulação. Por exemplo, no experimento com 100 requisições, cada computador realiza 20 requisições por segundo.

A validação da abordagem foi divida em duas partes. Na primeira parte, denominada de SM, efetuou-se a validação num ambiente sem o uso do mecanismo de controle de atraso. Desta forma, o processamento das requisições feitas pelos clientes é efetuado exclusivamente pelo *agente Fog*. Assim, todo o gerenciamento e encaminhamento dos dados ocorre na *Fog*. Na segunda parte, denominada de CM, efetuou-se a validação da abordagem com a utilização do mecanismo de controle de atraso. Neste cenário utilizou-se em conjunto *Fog* e Nuvem. O mecanismo de controle de atraso direciona o processamento das requisições para nuvem quando o *agente Fog* estiver sobrecarregado. Desta forma, o mecanismo diminui o estresse sobre o *hardware* do *agente Fog* e, consequentemente, reduz a latência da rede. Em ambos os cenários parte-se do princípio que o cliente conheça a localização do objeto que deseja consumir os dados.

5.2 Métricas

Na avaliação da eficiência da abordagem proposta foram empregadas duas métricas. Essas métricas de eficiência são definidas em latência, variação da latência e vazão. A seguir a descrição das métricas utilizadas:

Latência: a latência representa o tempo decorrido (t_r) em milissegundos. O (t_r) equivale a diferença temporal ocorrida entre o momento de realização de uma requisição pelo cliente (t_{req}) e a recepção da resposta correspondente (t_{rep}). A latência média (l_m) corresponde à média dos tempos de reposta observados em todos os n clientes, ou seja:

$$l_m = \frac{\sum_1^n t_r(i)}{n} = \frac{\sum_1^n t_{rep}(i) - t_{req}(i)}{n} \tag{5.1}$$

Variação da latência: mensura a variação do tempo decorrido em milissegundos entre as requisições. Essa flutuação do atraso de encaminhamento entre as requisições está expressa na Equação 5.2, onde (j_r) representa a diferença de tempo decorrido entre a requisição decorrente ($t_r(i)$) e a requisição antecedente ($t_r(i-1)$). A variação da latência média (j_m) corresponde à média da variação de tempo observada em todos os n clientes, ou seja:

$$j_m = \frac{\sum_1^n j_r(i)}{n} = \frac{\sum_1^n |t_r(i) - t_r(i-1)|}{n} \tag{5.2}$$

Vazão: (bw_r) representa o número de *bytes* transmitidos em um período de tempo. A vazão média (bw_m) corresponde à média da vazão observada em todas as (s) rodadas de simulação para cada quantidade de requisições por segundo, ou seja:

$$bw_m = \frac{\sum_1^s bw_r(i)}{s} \tag{5.3}$$

5.3 Resultados

Esta seção apresenta os resultados obtidos no ambiente de simulação. A validação da abordagem foi divida em duas partes. Na primeira parte, efetuou-se a validação num ambiente sem o uso do mecanismo de controle de atraso. Na segunda parte, realizou-se a validação da abordagem com a utilização do mecanismo de controle de atraso.

5.3.1 Cenário Sem o Mecanismo - SM

Esta subseção apresenta os resultados obtidos no cenário sem o mecanismo de controle de atraso. Neste cenário as requisições geradas pelos clientes são processadas somente na *Fog*. Desta forma, o *agente Fog* processa e encaminha todos os dados requisitados pelo cliente. Os resultados obtidos na simulação são apresentados em gráficos que realizam a comparação entre o serviço em nuvem (CC) e o serviço no *agente Fog* (Fog).

A Figura 5.2 apresenta a média da latência experimentada por todos os clientes ao requisitarem dados de um determinado sensor. O eixo Y representa em função de $log(n)$ a latência média em milissegundos (ms) e o eixo X apresenta o número de requisições por segundo (rps). Os resultados obtidos mostram que na *Fog* a latência entre 1 e 400 rps mantém-se inferior aos 100 ms. Entre os 1400 e 1800 rps a latência experimentada, praticamente, equiparam-se entre *Fog* e CC. No entanto, a partir das 2000 rps a latência do *Fog* ultrapassa os valores obtidos no CC. Esses valores altos no *Fog* ocorrem devido ao número elevado de requisições por segundo e a pouca quantidade de recursos computacionais existentes no *agente Fog*.

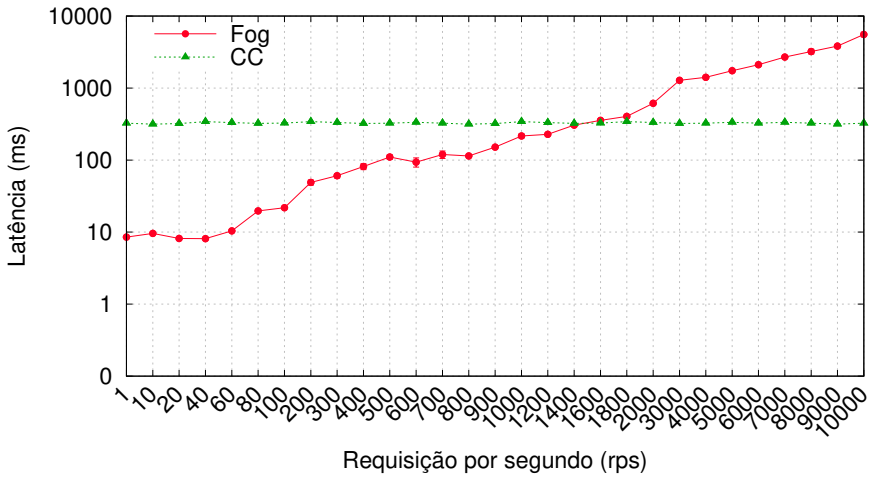


Figura 5.2: Latência

A Figura 5.3 apresenta a média da variação da latência experimentada por todos os clientes. O eixo Y representa em função de $log(n)$ a média da variação da latência em milissegundos (ms) e o eixo X apresenta o número de requisições por segundo (rps). O resultado mostra que entre 1 e 40 rps o *Fog* obteve variação inferior a 1 ms. No entanto, a partir dos 200 rps, a *Fog* atinge valores superiores aos obtidos no CC. Essa variação elevada pode ser vista, principalmente, nas 10000 rps, pois, o cliente experimenta latência que variam, aproximadamente, entre 300 milissegundos e 7,8 segundos.

A Figura 5.4 apresenta o número de *bytes* transmitidos em um período de tempo. O eixo Y representa em função de $log(n)$ a média da vazão em quilobytes por segundos (kB/s) e o eixo X apresenta o número de requisições por segundo (rps). O resultado obtido mostra que

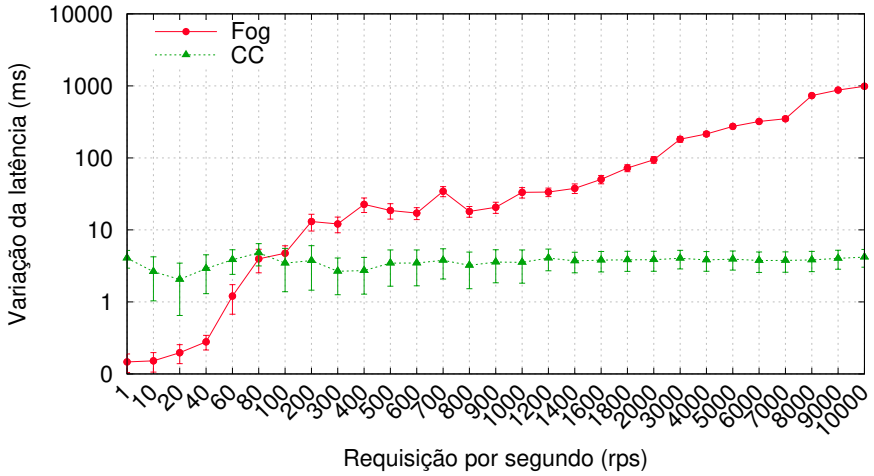


Figura 5.3: Variação da latência

por estar na borda da rede o *agente Fog* pode alcançar uma maior vazão. Isso pode ser visto no instante de 1200 rps em diante, onde a *Fog* obtêm uma maior vazão que o CC.

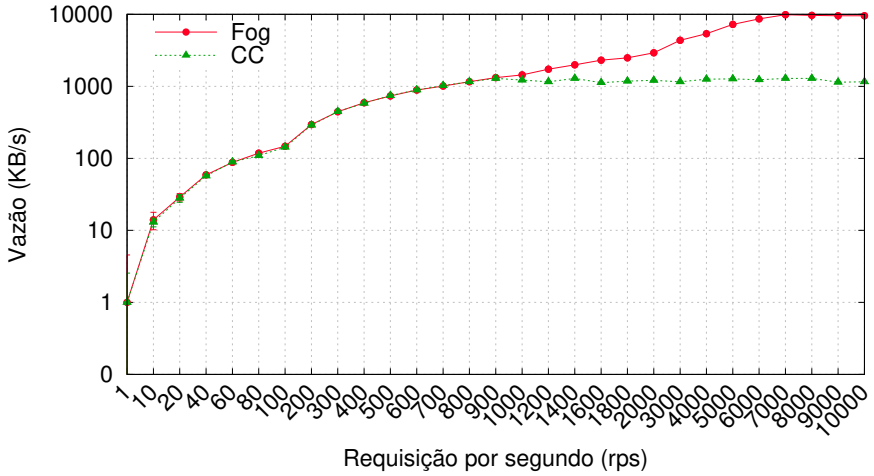


Figura 5.4: Vazão

A Figura 5.5 ilustra a média do consumo de recursos do *agente Fog*. O eixo Y representa o percentual de recursos utilizados e o eixo X apresenta o número de requisições por segundo (rps). O resultado obtido mostra que o consumo de memória RAM sofre poucas alterações. Isso pode ser visto entre 1 e 1800 rps, onde o consumo de memória mantêm-se próximo dos 10%. A utilização de memória somente altera-se apos as 2000 rps. No entanto, o percentual não ultrapassa os 30% de consumo. Diferente da memória, a CPU foi mais exigida. O percentual de consumo da CPU cresceu exponencialmente e alcançou os 85%.

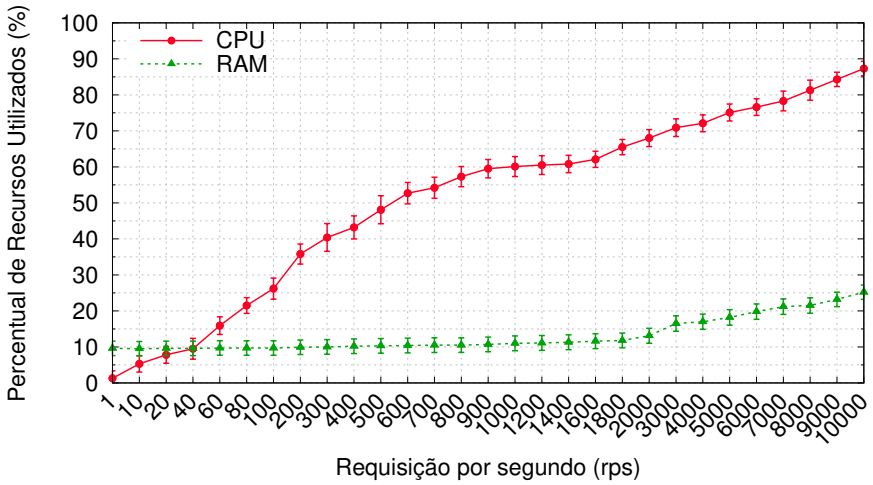


Figura 5.5: Consumo do *hardware*

5.3.2 Cenário Com o Mecanismo - CM

Esta subseção apresenta os resultados obtidos no cenário com o mecanismo de controle de atraso. Neste cenário as requisições geradas pelos clientes são processadas no *Fog*, mas quando houver uma sobrecarga de requisições as solicitações são direcionadas para a nuvem. Os resultados obtidos na simulação são apresentados em gráficos que realizam a comparação entre o serviço em nuvem (CC) e o serviço no *agente Fog* (Fog).

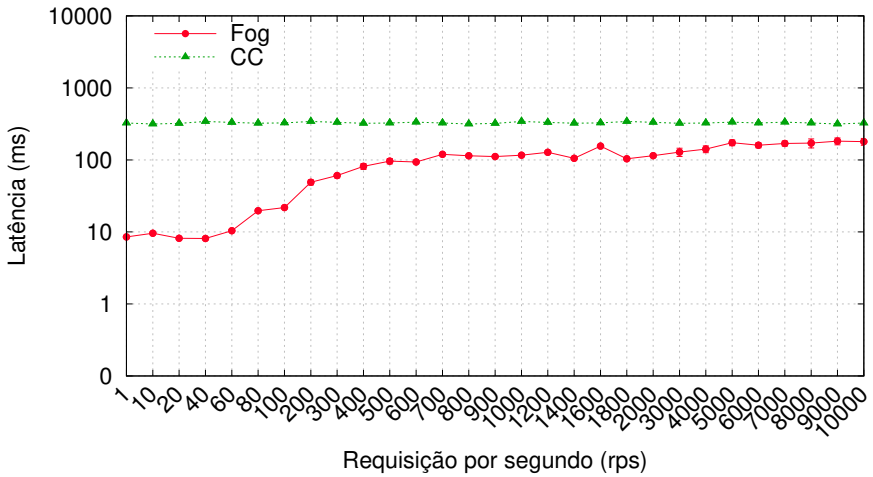


Figura 5.6: Latência

A Figura 5.6 apresenta a média da latência experimentada por todos os clientes ao requisitarem dados de um determinado sensor. O eixo Y representa em função de $log(n)$ a latência média em milissegundos (ms) e o eixo X apresenta o número de requisições por segundo (rps). Os resultados obtidos mostram que na *Fog* a latência entre 1 e 400 rps mantêm-se inferior aos 100 ms. Nas demais requisições por segundo, a latência experimentada pelos clientes ficam inferior aos resultados obtidos na CC. NeA Figura 5.6 apresenta a média da latência experimentada por todos os clientes ao requisitarem dados de um determinado sensor. O eixo Y representa em função de $log(n)$ a latência média em milissegundos (ms) e o eixo X apresenta o número de requisições por segundo (rps). Os resultados obtidos mostram que na *Fog* a latência entre 1 e 400 rps mantém-se inferior aos 100 ms. Nas demais requisições por segundo, a latência

experimentada pelos clientes ficam inferiores aos resultados obtidos na CC. Nestes cenários, devido a grande quantidade de requisições simultâneas o mecanismo direciona o cliente para a nuvem quando a latência no agente Fog for superior ao tempo de resposta da CC. Desta forma, o mecanismo evita latência superior ao tempo de resposta da CC e diminui a sobrecarga no agente Fog.

A Figura 5.7 apresenta a média da variação da latência experimentada por todos os clientes. O eixo Y representa em função de $log(n)$ a média da variação da latência em milissegundos (ms) e o eixo X apresenta o número de requisições por segundo (rps). O resultado mostra que entre 1 e 40 rps o *Fog* obteve variação inferior a 1 ms. No entanto, a partir dos 200 rps, a *Fog* atinge valores superiores aos obtidos no CC. Essa variação elevada pode ser vista, principalmente, nas 10000 rps, pois, o cliente experimenta latência que variam, aproximadamente, entre 300 milissegundos e 7,8 segundos.stes cenários, devido a grande quantidade de requisições simultâneas o mecanismo direciona o cliente para a nuvem quando a latência no agente Fog for superior ao tempo de resposta da CC. Desta forma, o mecanismo evita latências superiores ao tempo de resposta da CC e diminui a sobrecarga no agente Fog.

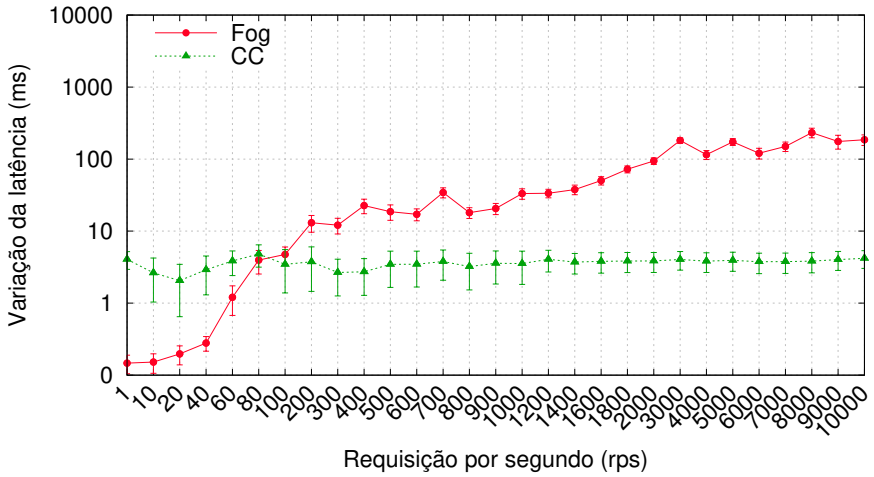


Figura 5.7: Variação da latência

A Figura 5.8 apresenta o número de *bytes* transmitidos em um período de tempo. O eixo Y representa em função de $log(n)$ a média da vazão em quilobytes por segundos (kB/s) e o eixo X apresenta o número de requisições por segundo (rps). O resultado obtido mostra que por estar na borda da rede o *agente Fog* pode alcançar uma maior vazão. Isso pode ser visto no instante de 1200 rps em diante, onde a *Fog* obtêm uma maior vazão que o CC.

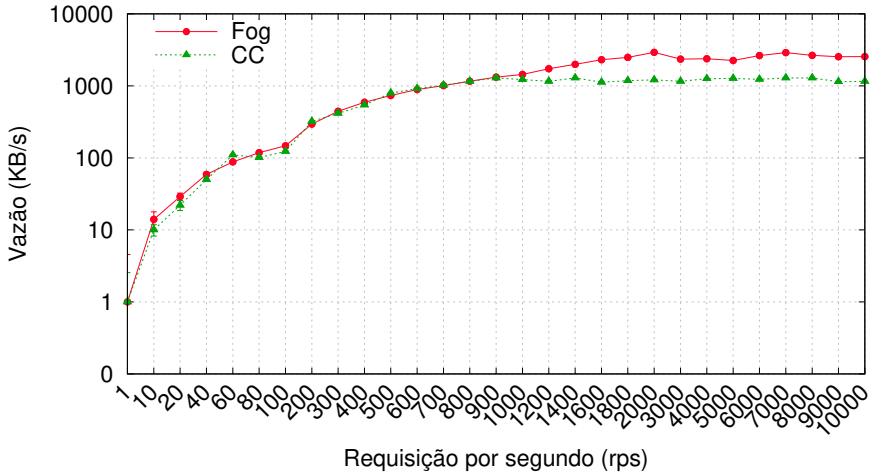


Figura 5.8: Vazão

A Figura 5.9 ilustra a média do consumo de recursos do *agente Fog*. O eixo Y representa o percentual de recursos utilizados e o eixo X apresenta o número de requisições por segundo (rps). O resultado obtido mostra que o consumo de memória RAM sofre poucas alterações. Isso pode ser visto entre 1 e 1800 rps, onde o consumo de memória mantêm-se próximo dos 10%. A utilização de memória somente altera-se apos as 2000 rps. No entanto, o percentual não ultrapassa os 25% de consumo. Diferente da memória, a CPU foi mais exigida. O percentual de consumo da CPU cresceu exponencialmente e alcançou os 68%.

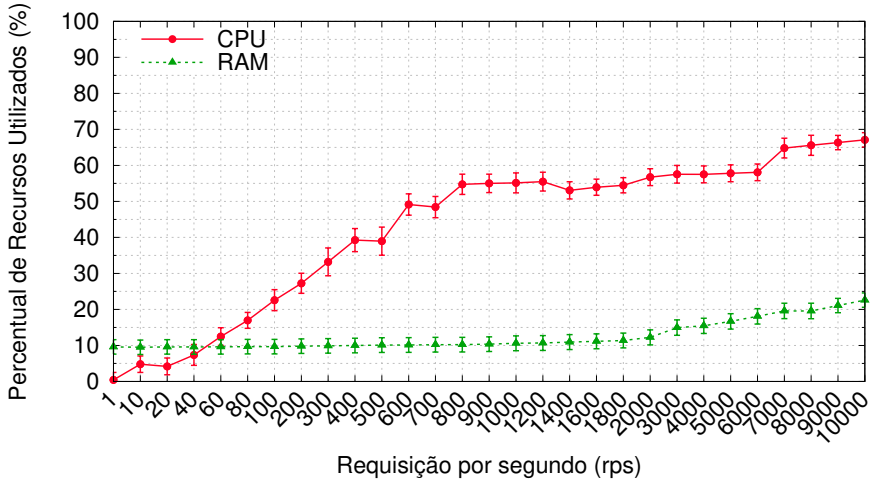


Figura 5.9: Consumo do *hardware*

5.3.3 Comparação entre os Cenários

A comparação entre os cenários sem e com o mecanismos de controle de atraso no cliente foi feita por meio da latência, variação da latência, vazão e o consumo de recursos de *hardware*. Para comparar a latência efetuamos o cálculo da média da latência experimentada pelo cliente em cada cenário com e sem o mecanismo. A comparação da média da variação da latência seguiu o mesmo calculo de média da latência. A vasão representa o número de *bytes* transmitidos em um período de tempo O consumo de *hardware* é o percentual de recursos computacionais utilizados em cada cenário.

A Figura 5.10 apresenta a comparação das métricas latência, variação da latência e vasão entre os dois cenários, SM e CM. O eixo Y representa a média de cada métrica e o eixo X apresenta o número de requisições por segundo (rps). Além disso, tanto Y quanto X estão representados em função de $log(n)$. A Figura 5.10(a) apresenta a comparação da latência em milissegundos, experimentada pelos clientes em cada senários. Nos cenários com 1, 10 e 100 rps tanto SM quanto CM possuem distinção mínima. A partir das 100 rps, o CM começa a obter uma pequena vantagem sobre SM. Os benefícios na utilização do mecanismo ficam mais claros nas 1000 e 10000 rps. A diferença no valor médio nessa quantidade de requisições ocorreu devido ao direcionamento das requisições para a nuvem. Assim, a latência em CM se manteve com valor médio inferior aos 300 milissegundos.

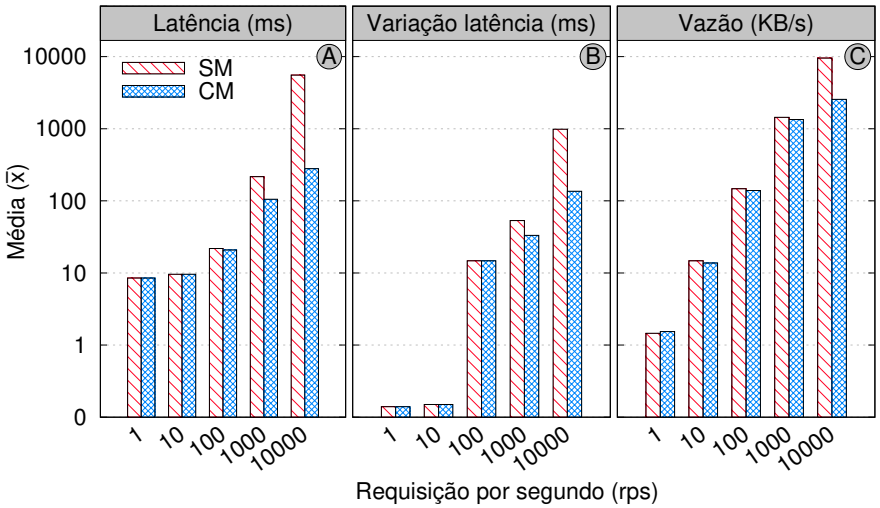


Figura 5.10: Comparação entre os cenários

Na Figura 5.10(b) contém os valores médios da variação da latência em milissegundos, experimentada pelos clientes nos cenários SM e CM. Com 1, 10 e 100 rps tanto SM quanto CM possuem distinção mínima. Com 1000 e 10000 rps os benefícios de uso do mecanismo ficam mais claros. A diferença no valor médio da variação da latência nos cenários com 1000 e 10000 clientes ocorreu devido ao direcionamento das requisições para a nuvem. Assim, a variação da latência em CM manteve-se inferior. A Figura 5.10(c) apresenta os valores obtidos da vazão. Com 1, 10 e 100 rps tanto SM quanto CM possuem distinção mínima. A distinção entre os dois cenários fica mais clara com 10000 rps.

A Figura 5.11 apresenta a comparação de consumo de recursos de *hardware* obtidos durante a simulação de cada cenário. Esse percentual do consumo foi obtido através de uma aplicação desenvolvida em *python*, com o uso da biblioteca *psutil*. Nos cenários com 1, 10 e 100 clientes o percentual de consumo de tanto da CPU quanto memória RAM teve uma diferenciação mínima. No entanto, nos cenários com 1000 e 10000 clientes, o CM diminui o consumo de CPU e memória RAM do *raspberry*. Essa redução no consumo de CPU ocorreu devido ao cliente direcionar suas requisições para a nuvem quando a latência no *agente Fog* estiver alta.

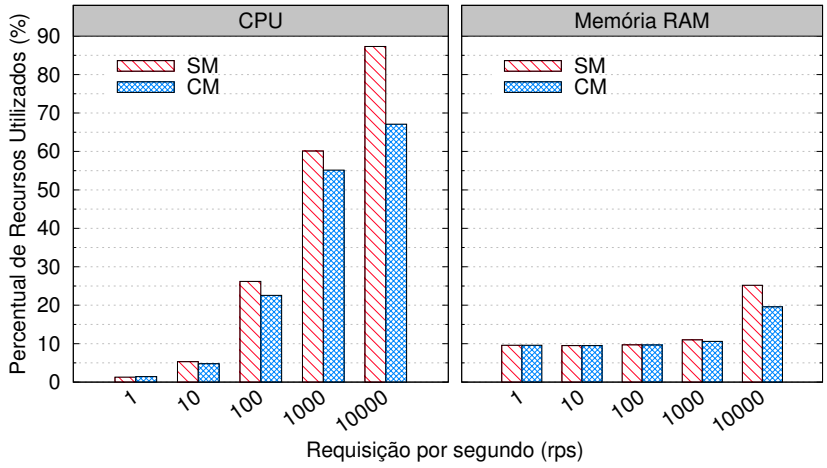


Figura 5.11: Comparação consumo do *hardware*

5.4 Resumo

Este capítulo apresentou a avaliação da abordagem para consumo de dados na borda da rede. Inicialmente foi apresentado o modelo da simulação. Nesta seção, foi exposto como foi desenvolvida e executada cada entidade da simulação, especificando todos os dispositivos utilizados no processo. Posteriormente, as métricas empregadas da validação foram apresentadas. Por fim, a última seção apresentou os resultados obtidos durante a simulação. Nos cenários de testes sem o mecanismo de controle de atraso no cliente, o aumento na quantidade de clientes simultâneos fez o tempo de resposta do *agente Fog* chegar a valores que não condizem com a latência esperada na borda da rede. Nos cenários com o uso do mecanismo de controle proporcionou uma melhora na latência. Contudo, por estar direcionando os clientes para a nuvem, a variação da latência foi maior.

Capítulo 6

Considerações Finais

Os dados desempenham papel importante na internet das coisas. Essa importância ocorre devido a grande quantidade de dados produzidos por todas as coisas do mundo real. Assim, o gerenciamento destes dados torna-se relevante para a IoT. Além disso, a grande quantidade de dispositivos da IoT aumenta o volume de dados gerados na borda da rede.Portanto, o gerenciamento dos dados possibilita uma melhora no armazenamento, processamento e análise do grande volume de dados gerados na IoT. Desta forma, o desenvolvimento de soluções eficientes para o gerenciamento de dados da IoT fá-se necessário.

Este trabalho apresentou uma abordagem de gerenciamento de dados da IoT na Fog. Essa abordagem utilizou os dispositivos contidos na borda rede para encaminhar os dados coletados na IoT para os cliente (consumidor). O cliente busca no indexador contido na nuvem o endereço do dispositivo contido na Fog que possua os dados do sensor escolhido. Após encontrar a localização, o cliente consome os dados exclusivamente na Fog sem precisar ir até a nuvem. No entanto, com um número elevado de clientes consumindo dados de um único dispositivo da Fog tende a estressar o *hardware* e aumentar a latência. Para minimizar o estresse no dispositivo, foi proposto um mecanismo simples implementa no cliente que efetua o controle da latência.

Um ambiente de simulação composto por uma combinação de dispositivos reais e virtuais foi projetado e desenvolvido para validar a eficiência da abordagem proposta neste trabalho. Os dispositivos reais foram utilizados com o intuito de fornecer maior grau de realidade na perspectiva dos dispositivos. Enquanto isso, a adoção concomitante de dispositivos virtualizados possibilitou a realização de experimentos com número maior de dispositivos e, consequentemente, aumentar o grau de realidade na perspectiva da rede. Por meio dessas simulações, foi possível avaliar as métricas propostas. Nas métricas latência e variação da latência, os resultados demonstraram que a abordagem proposta em cenários com no máximo 100 rps consegue manter um baixo tempo de resposta. No entanto, em cenários maiores, como 1000 rps, o tempo de resposta ficam superiores à latência existente em solicitações para a nuvem. Assim, a abordagem precisou da utilização de um mecanismo que consigo minimizar a latência experimentada por um determinado cliente. Esse aumento na latência ocorre devido à quantidade maior de requisições simultâneas que *agente Fog* precisa responder. Desta forma, a inserção do mecanismo de controle do atraso na abordagem permitiu minimizar o tempo de resposta do *agente Fog*. Outro ponto importa, as simulações conseguiram apresentar o consumo de *hardware*. Os resultados obtidos mostraram até que ponto do dispositivo embardo aguenta processar e encaminhar os dados na *Fog*.

Como trabalhos futuros, pretende-se utilizar ambientes complexos, outra base de dados, outros modelos de embarcados e alterar a filtragem dos dados. Para validar melhor a proposta, pretende-se realizar teste em ambientes mais complexos, com uma maior quantidade de nós. Outro ponto importante, efetuar alteração da base de dados por uma base mais leve

computacionalmente. Nos testes utilizou-se o mongoDB, uma base que exige um pouco mais de recursos computacionais. A *Fog* é composta com dispositivos heterogêneos. Desta forma, para uma melhor validação, pretende-se utilizar outros modelos de embarcados no ambiente de simulação. Além disso, alterar o método de filtragem dos dados para filtrar por distribuição geográfica. Atualmente, os dados são filtrados através do nome do sensor. Por fim, manter todos os dados coletados na borda da rede e transferir o mecanismo de controle de atraso do cliente para o *agente Fog*. Ao manter os dados na borda da rede, aumentaria a privacidade dos dados. Pois, os dados seriam gerenciados e disponibilizados por seu próprio dono, sem a necessidade de ter um terceiro envolvido. No entanto, como visto nos experimentos, com uma grande quantidade de clientes o dispositivo da *Fog* tende a sobrecarregar. Desta forma, para evitar esta sobrecarga será implementado um mecanismo baseado no consumo dos recursos existentes no dispositivo. O mecanismo vai identificar o sensor mais acessado, encaminhar os dados deste sensor para a nuvem e direcionar os clientes para a nuvem. Assim, somente será enviado os dados para a nuvem quando o dispositivo da Fog estiver sobrecarregado.

Referências Bibliográficas

[Aazam e Huh, 2014] Aazam, M. e Huh, E. N. (2014). Fog computing and smart gateway based communication for cloud of things. Em *2014 International Conference on Future Internet of Things and Cloud*, páginas 464–470. doi:10.1109/FiCloud.2014.83.

[Aazam et al., 2014] Aazam, M., Khan, I., Alsaffar, A. A. e Huh, E. N. (2014). Cloud of things: Integrating internet of things and cloud computing and the issues involved. Em *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences Technology (IBCAST) Islamabad, Pakistan, 14th - 18th January, 2014*, páginas 414–419. ISSN: 2151-1403. doi: 10.1109/IBCAST.2014.6778179.

[Aggarwal et al., 2013] Aggarwal, C. C., Ashish, N. e Sheth, A. (2013). *The Internet of Things: A Survey from the Data-Centric Perspective*, páginas 383–428. Springer US, Boston, MA. ISBN: 978-1-4614-6309-2. doi: 10.1007/978-1-4614-6309-2_12. url: https://doi.org/10.1007/978-1-4614-6309-2_12.

[Amaral et al., 2016] Amaral, L. A., de Matos, E., Tiburski, R. T., Hessel, F., Lunardi, W. T. e Marczak, S. (2016). *Middleware Technology for IoT Systems: Challenges and Perspectives Toward 5G*, páginas 333–367. Springer International Publishing, Cham. ISBN: 978-3-319-30913-2. doi: 10.1007/978-3-319-30913-2_15. url: https://doi.org/10.1007/978-3-319-30913-2_15.

[Amarni, 2017] Amarni, B. (2017). Pi64. https://github.com/bamarni/pi64. Acessado em 10/06/2017.

[Amazon, 2017] Amazon (2017). Amazon EC2. https://aws.amazon.com/pt/ec2/. Acessado em 10/06/2017.

[Arduino, 2017] Arduino (2017). Arduino. https://www.arduino.cc/. Acessado em 12/06/2017.

[Arkessa, 2017] Arkessa (2017). Arkessa. http://www.arkessa.com/. Acessado em 08/03/2017.

[Armbrust et al., 2010] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. e Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53(4):50–58. ISSN: 0001-0782. doi: 10.1145/1721654.1721672. url: http://doi.acm.org/10.1145/1721654.1721672.

[Atzori et al., 2010] Atzori, L., Iera, A. e Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805. ISSN: 1389-1286. doi: http://dx.doi.org/10.1016/j.comnet.2010.05.010 url: http://www.sciencedirect.com/science/article/pii/S1389128610001568.

[Bilal et al., 2014] Bilal, K., Malik, S. U. R., Khan, S. U. e Zomaya, A. Y. (2014). Trends and challenges in cloud datacenters. *IEEE Cloud Computing*, 1(1):10–20. ISSN: 2325-6095. doi: 10.1109/MCC.2014.26.

[Bonomi et al., 2014] Bonomi, F., Milito, R., Natarajan, P. e Zhu, J. (2014). *Fog Computing: A Platform for Internet of Things and Analytics*, páginas 169–186. Springer International Publishing, Cham. ISBN: 978-3-319-05029-4. doi: 10.1007/978-3-319-05029-4_7. url: https://doi.org/10.1007/978-3-319-05029-4_7.

[Bonomi et al., 2012] Bonomi, F., Milito, R., Zhu, J. e Addepalli, S. (2012). Fog computing and its role in the internet of things. Em *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, páginas 13–16, New York, NY, USA. ACM. ISBN: 978-1-4503-1519-7. doi: 10.1145/2342509.2342513. url: http://doi.acm.org/10.1145/2342509.2342513.

[Botta et al., 2016] Botta, A., de Donato, W., Persico, V. e Pescapé, A. (2016). Integration of cloud computing and internet of things: A survey. *Future Generation Computer Systems*, 56:684 – 700. ISSN: 0167-739X. doi: http://dx.doi.org/10.1016/j.future.2015.09.021. url: http://www.sciencedirect.com/science/article/pii/S0167739X15003015.

[Calbimonte et al., 2014] Calbimonte, J.-P., Sarni, S., Eberle, J. e Aberer, K. (2014). Xgsn: an open-source semantic sensing middleware for the web of things. Em *Joint Proceedings of the 6th International Workshop on the Foundations, Technologies and Applications of the Geospatial Web, TC*, páginas 51–66. url: http://ceur-ws.org/Vol-1401/#paper-04.

[Cao et al., 2014] Cao, J., Ma, M., Li, H., Zhang, Y. e Luo, Z. (2014). A survey on security aspects for lte and lte-a networks. *IEEE Communications Surveys Tutorials*, 16(1):283–302. ISSN: 1553-877X. doi: 10.1109/SURV.2013.041513.00174.

[Carriots, 2017] Carriots (2017). Carriots. https://www.carriots.com/. Acessado em 08/03/2017.

[Chang, 2014] Chang, K. H. (2014). Bluetooth: a viable solution for iot? [industry perspectives]. *IEEE Wireless Communications*, 21(6):6–7. ISSN: 1536-1284. doi: 10.1109/MWC.2014.7000963.

[Cheng et al., 2015] Cheng, B., Papageorgiou, A., Cirillo, F. e Kovacs, E. (2015). Geelytics: Geo-distributed edge analytics for large scale iot systems based on dynamic topology. Em *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, páginas 565–570. doi: 10.1109/WF-IoT.2015.7389116.

[Chunli, 2012] Chunli, L. (2012). Intelligent transportation based on the internet of things. Em *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, páginas 360–362. doi: 10.1109/CECNet.2012.6201865.

[Coskun et al., 2013] Coskun, V., Ozdenizci, B. e Ok, K. (2013). A survey on near field communication (nfc) technology. *Wireless Personal Communications*, 71(3):2259–2294. ISSN: 1572-834X. doi: 10.1007/s11277-012-0935-5. url: https://doi.org/10.1007/s11277-012-0935-5.

[Edmonds et al., 2015] Edmonds, O.-W. A., Papaspyrou, A. e Metsch, T. (2015). Open cloud computing interface-core. vol. GFD.183, Abril 2015. url: https://redmine.ogf.org/attachments/177/core.pdf.

[Farhady et al., 2015] Farhady, H., Lee, H. e Nakao, A. (2015). Software-defined networking: A survey. *Computer Networks*, 81:79 – 95. ISSN: 1389-1286. doi: http://dx.doi.org/10.1016/j.comnet.2015.02.014. url: http://dx.doi.org/10.1016/j.comnet.2015.02.014.

[Fernandez e Pallis, 2014] Fernandez, F. e Pallis, G. C. (2014). Opportunities and challenges of the internet of things for healthcare: Systems engineering perspective. Em *2014 4th International Conference on Wireless Mobile Communication and Healthcare - Transforming Healthcare Through Innovations in Mobile and Wireless Technologies (MOBIHEALTH)*, páginas 263–266. doi: 10.1109/MOBIHEALTH.2014.7015961.

[Ferro e Potorti, 2005] Ferro, E. e Potorti, F. (2005). Bluetooth and wi-fi wireless protocols: a survey and a comparison. *IEEE Wireless Communications*, 12(1):12–26. ISSN: 1536-1284 doi: 10.1109/MWC.2005.1404569.

[Fiware, 2016] Fiware (2016). Internet of Things (IoT) Services Enablement Architecture. http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Internet_of_Things_%28IoT%29_Services_Enablement_Architecture. Acessado em 30/06/2016.

[Fiware, 2017a] Fiware (2017a). Documentation Orion ContextBroker. http://fiware-orion.readthedocs.io/en/master/admin/index.html. Acessado em 08/03/2017.

[Fiware, 2017b] Fiware (2017b). Fiware. https://www.fiware.org/. Acessado em 08/03/2017.

[Fiware, 2017c] Fiware (2017c). Fiware Device Simulator documentation. https://fiware-device-simulator.readthedocs.io/en/latest/. Acessado em 08/03/2017.

[Gazis et al., 2015] Gazis, V., Leonardi, A., Mathioudakis, K., Sasloglou, K., Kikiras, P. e Sudhaakar, R. (2015). Components of fog computing in an industrial internet of things context. Em *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking - Workshops (SECON Workshops)*, páginas 1–6. doi: 10.1109/SECONW.2015.7328144.

[Google, 2017] Google (2017). Google Cloud Platform. https://cloud.google.com/?hl=pt-br. Acessado em 10/06/2017.

[Gubbi et al., 2013] Gubbi, J., Buyya, R., Marusic, S. e Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services & Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond. ISSN: 0167-739X. doi: http://dx.doi.org/10.1016/j.future.2013.01.010. url: http://www.sciencedirect.com/science/article/pii/S0167739X13000241.

[Gungor e Hancke, 2009] Gungor, V. C. e Hancke, G. P. (2009). Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *IEEE Transactions on Industrial Electronics*, 56(10):4258–4265. ISSN: 4258-4265. doi: 10.1109/TIE.2009.2015754.

[He et al., 2014] He, W., Yan, G. e Xu, L. D. (2014). Developing vehicular data cloud services in the iot environment. *IEEE Transactions on Industrial Informatics*, 10(2):1587–1595. ISSN 1551-3203. doi: 10.1109/TII.2014.2299233.

[Ishaq et al., 2013] Ishaq, I., Carels, D., Teklemariam, G. K., Hoebeke, J., Abeele, F. V. d., Poorter, E. D., Moerman, I. e Demeester, P. (2013). Ietf standardization in the field of the internet of things (iot): A survey. *Journal of Sensor and Actuator Networks*, 2(2):235–287. ISSN: 2224-2708. doi: 10.3390/jsan2020235. url: http://www.mdpi.com/2224-2708/2/2/235.

[Jararweh et al., 2015] Jararweh, Y., Al-Ayyoub, M., Darabseh, A., Benkhelifa, E., Vouk, M. e Rindos, A. (2015). Sdiot: a software defined based internet of things framework. *Journal of Ambient Intelligence and Humanized Computing*, 6(4):453–461. ISSN: 1868-5145. doi: 10.1007/s12652-015-0290-y. url: https://doi.org/10.1007/s12652-015-0290-y.

[Jia et al., 2012] Jia, X., Feng, Q., Fan, T. e Lei, Q. (2012). Rfid technology and its applications in internet of things (iot). Em *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, páginas 1282–1285. doi: 10.1109/CECNet.2012.6201508.

[Karagiannis et al., 2015] Karagiannis, V., Chatzimisios, P., Vazquez-Gallego, F. e Alonso-Zarate, J. (2015). A survey on application layer protocols for the internet of things. *Transaction on IoT and Cloud Computing*, 3(1):11–17. ISSN: 2331-4761.

[Khan et al., 2012] Khan, R., Khan, S. U., Zaheer, R. e Khan, S. (2012). Future internet: The internet of things architecture, possible applications and key challenges. Em *2012 10th International Conference on Frontiers of Information Technology*, páginas 257–260. doi: 10.1109/FIT.2012.53.

[Khanafer et al., 2014] Khanafer, M., Guennoun, M. e Mouftah, H. T. (2014). A survey of beacon-enabled ieee 802.15.4 mac protocols in wireless sensor networks. *IEEE Communications Surveys Tutorials*, 16(2):856–876. ISSN: 1553-877X. doi: 10.1109/SURV.2013.112613.00094.

[Kostelnik et al., 2011] Kostelnik, P., Sarnovsk, M. e Furdik, K. (2011). The semantic middleware for networked embedded systems applied in the internet of things and services domain. *Scalable Computing: Practice and Experience*, 12(3):307–316. ISSN: 1895-1767.

[Kurose e Ross, 2012] Kurose, J. F. e Ross, K. W. (2012). *Computer Networking: A Top-Down Approach (6th Edition)*. Pearson, 6th edition. ISSN: 0132856204, 9780132856201.

[Li et al., 2015] Li, S., Xu, L. D. e Zhao, S. (2015). The internet of things: a survey. *Information Systems Frontiers*, 17(2):243–259. ISSN: 1572-9419. doi: 10.1007/s10796-014-9492-7. url: https://doi.org/10.1007/s10796-014-9492-7.

[LinkSmart, 2016] LinkSmart (2016). LinkSmart. https://docs.linksmart.eu/. Acessado em 01/08/2016.

[Mahmud e Buyya, 2016] Mahmud, R. e Buyya, R. (2016). Fog computing: A taxonomy, survey and future directions. *CoRR*, abs/1611.05539. url: http://arxiv.org/abs/1611.05539.

[Mell e Grance, 2011] Mell, P. M. e Grance, T. (2011). Sp 800-145. the nist definition of cloud computing. National Institute of Standards & Technology. Gaithersburg, MD, United States.

[Mineraud et al., 2016] Mineraud, J., Mazhelis, O., Su, X. e Tarkoma, S. (2016). A gap analysis of internet-of-things platforms. *Computer Communications*, 89:5 – 16. Internet of Things Research challenges and Solutions. ISSN: 0140-3664. doi: http://dx.doi.org/10.1016/j.comcom.2016.03.015. url: http://www.sciencedirect.com/science/article/pii/S0140366416300731.

[Minerva et al., 2015] Minerva, R., Biru, A. e Rotondi, D. (2015). Towards a definition of the internet of things (iot). *IEEE Internet Initiative, Torino, Italy*.

[Miorandi et al., 2012] Miorandi, D., Sicari, S., Pellegrini, F. D. e Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497 – 1516. ISSN: 2224-2708. doi: http://dx.doi.org/10.1016/j.adhoc.2012.02.016. url: http://www.sciencedirect.com/science/article/pii/S1570870512000674.

[Moltchanov e Rocha, 2014] Moltchanov, B. e Rocha, O. R. (2014). A context broker to enable future iot applications and services. Em *2014 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, páginas 263–268. ISSN: 2157-0221. doi: 10.1109/ICUMT.2014.7002113.

[MongoDB, 2017] MongoDB (2017). MongoDB. https://www.mongodb.com/what-is-mongodb. Acessado em 12/06/2017.

[Nastic et al., 2015] Nastic, S., Truong, H.-L. e Dustdar, S. (2015). Sdg-pro: a programming framework for software-defined iot cloud gateways. *Journal of Internet Services and Applications*, 6(1):21. ISBN: 1869-0238. doi: 10.1186/s13174-015-0037-1. url: https://doi.org/10.1186/s13174-015-0037-1.

[Niagara, 2017] Niagara (2017). Niagara. http://www.niagaraax.com/. Acessado em 08/03/2017.

[Nimbits, 2016] Nimbits (2016). Nimbits. http://www.nimbits.com/. Acessado em 01/08/2016.

[Nunes et al., 2014] Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K. e Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys Tutorials*, 16(3):1617–1634. ISSN: 1617-1634. doi: 10.1109/SURV.2014.012214.00180.

[OpenIoT, 2017] OpenIoT (2017). OpenIoT. http://www.openiot.eu/. Acessado em 08/03/2017.

[openSUSE, 2017] openSUSE (2017). opensuse. https://en.opensuse.org/HCL:Raspberry_Pi3. Acessado em 10/06/2017.

[Palo et al., 2013] Palo, A., Zuccaro, L., Simeoni, A., Suraci, V., Musto, L. e Garino, P. (2013). A common open interface to programmatically control and supervise open networks in the future internet. Em *2013 Future Network Mobile Summit*, páginas 1–9.

[Perera et al., 2014] Perera, C., Zaslavsky, A., Christen, P. e Georgakopoulos, D. (2014). Context aware computing for the internet of things: A survey. *IEEE Communications Surveys Tutorials*, 16(1):414–454. ISSN: 1553-877X. doi: 10.1109/SURV.2013.042313.00197.

[Prazeres e Serrano, 2016] Prazeres, C. e Serrano, M. (2016). Soft-iot: Self-organizing fog of things. Em *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, páginas 803–808. doi: 10.1109/WAINA.2016.153.

[Qin et al., 2016] Qin, Y., Sheng, Q. Z., Falkner, N. J., Dustdar, S., Wang, H. e Vasilakos, A. V. (2016). When things matter: A survey on data-centric internet of things. *Journal of Network and Computer Applications*, 64:137 – 153. ISSN: 1084-8045. doi: http://dx.doi.org/10.1016/j.jnca.2015.12.016. url: http://www.sciencedirect.com/science/article/pii/S1084804516000606.

[Qin et al., 2014] Qin, Z., Denker, G., Giannelli, C., Bellavista, P. e Venkatasubramanian, N. (2014). A software defined networking architecture for the internet-of-things. Em *2014 IEEE Network Operations and Management Symposium (NOMS)*, páginas 1–9. ISSN: 1542-1201. doi: 10.1109/NOMS.2014.6838365.

[Rahmani et al., 2015] Rahmani, A. M., Thanigaivelan, N. K., Gia, T. N., Granados, J., Negash, B., Liljeberg, P. e Tenhunen, H. (2015). Smart e-health gateway: Bringing intelligence to internet-of-things based ubiquitous healthcare systems. Em *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, páginas 826–834. ISSN: 2331-9852. doi: 10.1109/CCNC.2015.7158084.

[Raspberry, 2017a] Raspberry (2017a). Kernel source tree for Raspberry Pi Foundation-provided kernel builds. https://github.com/raspberrypi/linux. Acessado em 30/06/2017.

[Raspberry, 2017b] Raspberry (2017b). Raspberry Foundation. https://www.raspberrypi.org/. Acessado em 30/05/2017.

[Raspberry, 2017c] Raspberry (2017c). Raspberry Pi 3 Model B. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Acessado em 30/06/2017.

[Rawat et al., 2014] Rawat, P., Singh, K. D., Chaouchi, H. e Bonnin, J. M. (2014). Wireless sensor networks: a survey on recent developments and potential synergies. *The Journal of Supercomputing*, 68(1):1–48. ISSN: 1573-0484. doi: 10.1007/s11227-013-1021-9. url: https://doi.org/10.1007/s11227-013-1021-9.

[Rittinghouse e Ransome, 2009] Rittinghouse, J. e Ransome, J. (2009). *Cloud Computing: Implementation, Management, and Security*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition. ISBN: 1439806802, 9781439806807.

[Sarma et al., 2000] Sarma, S., Brock, D. L. e Ashton, K. (2000). The networked physical world. *Auto-ID Center White Paper MIT-AUTOID-WH-001*. http://cocoa.ethz.ch/downloads/2014/06/None_MIT-AUTOID-WH-001.pdf. Acessado em 13/07/2016.

[Sefraoui et al., 2012] Sefraoui, O., Aissaoui, M. e Eleuldj, M. (2012). Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3). http://research.ijcaonline.org/volume55/number3/pxc3882991.pdf. Acessado em 12/07/2016.

[Sheng et al., 2013] Sheng, Z., Yang, S., Yu, Y., Vasilakos, A. V., Mccann, J. A. e Leung, K. K. (2013). A survey on the ietf protocol suite for the internet of things: standards, challenges, and opportunities. *IEEE Wireless Communications*, 20(6):91–98. ISSN: 1536-1284. doi: 10.1109/MWC.2013.6704479.

[Shukla e Simmhan, 2017] Shukla, A. e Simmhan, Y. (2017). Benchmarking distributed stream processing platforms for iot applications. páginas 90–106. ISBN: 978-3-319-54334-5. doi: 10.1007/978-3-319-54334-5_7. url: https://doi.org/10.1007/978-3-319-54334-5_7.

[Silva et al., 2013] Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., Carvalho, A. C. P. L. F. d. e Gama, J. a. (2013). Data stream clustering: A survey. *ACM Comput. Surv.*, 46(1):13:1–13:31. ISSN: 0360-0300. doi: 10.1145/2522968.2522981. url: http://doi.acm.org/10.1145/2522968.2522981.

[Sofia, 2016a] Sofia (2016a). Feep IoT Platform Sofia2: technical view. http://sofia2.com/docs/Sofia2_IoT_Platform-Vista_Tecnica(nov_2016).pdf. Acessado em 01/12/2016.

[Sofia, 2016b] Sofia (2016b). Sofia. http://sofia2.com/. Acessado em 01/08/2016.

[Soldatos et al., 2015] Soldatos, J., Kefalakis, N., Hauswirth, M., Serrano, M., Calbimonte, J.-P., Riahi, M., Aberer, K., Jayaraman, P. P., Zaslavsky, A., Žarko, I. P., Skorin-Kapov, L. e Herzog, R. (2015). Openiot: Open source internet-of-things in the cloud. Em Podnar Žarko, I., Pripužić, K. e Serrano, M., editores, *Interoperability and Open-Source Solutions for the Internet of Things: International Workshop, FP7 OpenIoT Project, Held in Conjunction with SoftCOM 2014, Split, Croatia, September 18, 2014, Invited Papers*, páginas 13–25. Springer International Publishing, Cham. ISBN: 78-3-319-16546-2. doi: 10.1007/978-3-319-16546-2_3. url: https://doi.org/10.1007/978-3-319-16546-2_3.

[Soldatos et al., 2012] Soldatos, J., Serrano, M. e Hauswirth, M. (2012). Convergence of utility computing with the internet-of-things. Em *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, páginas 874–879. doi: 10.1109/IMIS.2012.135.

[Sukumar e Ravi, 2016] Sukumar, P. e Ravi, D. S. (2016). Iot based efficient vehicle location help line system using nfc. *International Journal of Emerging Technologies in Engineering Research (IJETER)*, 4(5):10–13. ISSN: 2454-6410.

[ThingSpeak, 2016a] ThingSpeak (2016a). https://www.mathworks.com/help/thingspeak/. Acessado em 01/08/2016.

[ThingSpeak, 2016b] ThingSpeak (2016b). ThingSpeak. https://thingspeak.com/. Acessado em 01/08/2016.

[Vacca, 2017] Vacca, J. R. (2017). *Cloud computing security: foundations and challenges*. CRC Press. ISBN: 978-1-4822-6094-6.

[Vasseur e Dunkels, 2010] Vasseur, J.-P. e Dunkels, A. (2010). *Interconnecting Smart Objects with IP: The Next Internet*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN: 0123751659, 9780123751652.

[Weiser, 1991] Weiser, M. (1991). The computer for the 21 st century. *Scientific American*, 265(3):94–105. ISSN: 00368733, 19467087. URL: http://www.jstor.org/stable/24938718. Acessado em 03/08/2016.

[Whitmore et al., 2015] Whitmore, A., Agarwal, A. e Da Xu, L. (2015). The internet of things—a survey of topics and trends. *Information Systems Frontiers*, 17(2):261–274. ISSN: 1572-9419. doi: 10.1007/s10796-014-9489-2. url: https://doi.org/10.1007/s10796-014-9489-2.

[Wu et al., 2015] Wu, D., Arkhipov, D. I., Asmare, E., Qin, Z. e McCann, J. A. (2015). Ubiflow: Mobility management in urban-scale software defined iot. Em *2015 IEEE Conference on Computer Communications (INFOCOM)*, páginas 208–216. ISSN: 0743-166X. doi: 10.1109/INFOCOM.2015.7218384.

[Xu et al., 2014] Xu, L. D., He, W. e Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243. ISSN: 1551-3203. doi: 10.1109/TII.2014.2300753.

[Yang et al., 2013] Yang, L., Yang, S. e Plotnick, L. (2013). How the internet of things technology enhances emergency response operations. *Technological Forecasting and Social Change*, 80(9):1854 – 1867. Planning and Foresight Methodologies in Emergency Preparedness and Management. ISSN: 0040-1625. doi: http://dx.doi.org/10.1016/j.techfore.2012.07.011. url: http://www.sciencedirect.com/science/article/pii/S0040162512001801.

[Yi et al., 2015] Yi, S., Li, C. e Li, Q. (2015). A survey of fog computing: Concepts, applications and issues. Em *Proceedings of the 2015 Workshop on Mobile Big Data*, Mobidata '15, páginas 37–42, New York, NY, USA. ACM. ISBN: 978-1-4503-3524-9. doi: 10.1145/2757384.2757397. url: http://doi.acm.org/10.1145/2757384.2757397.

[Zhang et al., 2010] Zhang, Q., Cheng, L. e Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18. ISSN: 1869-0238. doi: 10.1007/s13174-010-0007-6. url: https://doi.org/10.1007/s13174-010-0007-6.